EDK II DSC File Specification

May 2010
Revision 1.22

Acknowledgements

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

A license is hereby granted to copy and reproduce this specification for internal use only.

No other license, express or implied, by estoppel or otherwise, to any other intellectual property rights is granted herein.

Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

This specification is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

*Other names and brands may be claimed as the property of others.

Copyright ©2006 - 2010 Intel Corporation. All rights reserved.

Revision History

Revision	Revision History	Date
1.0	Initial release .	December 2007
1.1	Updated based on errata	August 2008
1.2	Updated based on enhancement requests	June 2009
1.21	Updated based o n errata and enhancement requests • Added language filters: RFC_LANGUAGES and ISO_LANGUAGES • Added Note that a reserved macro name, MDEPKG_NDEBUG • Definitions in DSC file are now global to both DSC and FDF files • PCD Values may be constructed using C-style expressions provided the result of the expression matches the datum type of the PCD • FeatureFlagExpression is now defined as a C-style expression using C relational, equality and logical numeric and bitwise operators and/or arithmetic and bitwise operators that evaluate to a value that matches the Datum Type of the PCD. Precedence and associativity follow C standards Spec changed to match exsting formats: • SUPPORTED_ARCHITECTURES and BUILD_TARGETS use the pipe “|” separator, not the comma character	March 2010
1.22	Errata and grammatical changes	May 2010

Contents

1
Introduction 1
1.1 Overview 1
1.2 Related Information 2
1.3 Terms 2
1.4 Conventions Used in this Document 5
1.4.1 Data Structure Descriptions 5
1.4.2 Pseudo-Code Conventions 5
1.4.3 Typographic Conventions 6

2
DSC Overview 9
2.1 Processing Overview 9
2.2 Build Description File Format 11
2.2.1 Section Entries 12
2.2.2 Comments 12
2.2.3 Valid Entries 12
2.2.4 Naming Conventions 12
2.2.5 !include Statement Processing 13
2.2.6 Macro Statements 13
2.2.7 Conditional Directive Statements (!if...) 13
2.2.8 [Defines] Section Processing 14
2.2.9 [BuildOptions] Section 16
2.2.10 [SkuIds] Section Processing 19
2.2.11 [Libraries] Section Processing 19
2.2.12 [LibraryClasses] Section Processing 20
2.2.13 PCD Sections Processing 21
2.2.14 [Components] Section Processing 25
2.2.15 [UserExtensions] Section 28

3
EDK II DSC Format Specification 29
3.1 Introduction 29
3.2 Platform DSC Definition 29
3.2.1 Common Definitions 30
3.2.2 MACRO Statements 31
3.2.3 Conditional Directive Statements 34
3.2.4 !include Statements 38
3.3 Header Section 38
3.4 [Defines] Section 40
3.5 [BuildOptions] Section 44
3.6 [SkuIds] Section 48
3.7 [Libraries] Section 49
3.8 [LibraryClasses] Section 50

3.9 PCD Sections 52
3.9.1 [Pedestrianisms] Section 53
3.9.2 [PcdsFixedAtBuild] Section 55
3.9.3 [PcdsPatchableInModule] Section 57
3.9.4 [PcdsDynamic] Section 59
3.9.5 [PcdsDynamicEx] Section 62
3.10 [Components] Section 66
3.11 [UserExtensions] Section 72
Appendix A
Variables 75
Appendix B
Sample EDK II DSC File 79
Appendix C
Module Types 91

Tables

Table 1. EDK Build Infrastructure Support Matrix 1
Table 2.EDK II [Defines] Section Elements 15
Table 3.EDK II [BuildOptions] Section Elements: Optional Tags 17
Table 4.EDK II [BuildOptions] Variable Descriptions 18
Table 5.Standard Variables 76
Table 6. EDK II Module Types 91

Figures

Figure 1. EDK II Parsing Data Flow 10
Figure 2. Developer Workstation Layout - EDK II Build 11

1 Introduction

1.1 Overview

This document describes the EDK II Platform Description file (DSC) format. The EDK Build Tools are included as part of the EDK II compatibility package. In order to use EDK II Modules or the EDK II Build Tools, an EDK II DSC and FDF file must be used.

EDK II tools use INI style text based files to describe components, platforms and firmware volumes. While similar to EDK DSC files, the EDK II DSC file format is different, and new utilities have been provided to process these files.

This design has the following goals.

Compatible

The EDK II DSC format does not support EDK DSC format, nor can EDK tools be used to parse the EDK II DSC format files. The EDK II DSC Format must maintain backward compatibility for supporting existing EDK INF file formats. This means that the changes made to this specification do not require changes for standard INF files.

Simplified platform build and configuration

One goal of this format is to simplify the build setup and configuration for a given platform. It was also designed to simplify the process of adding EDK II firmware components to a firmware volume on a given platform.

Table 1 describes the compatibility between platform, module and component builds.

Table 1. EDK Build Infrastructure Support Matrix

	EDK DSC	EDK II DSC	EDK FDF	EDK II FDF	EDK INF	EDK II INF
EDK Build Tools	YES	NO	YES	NO	YES	NO
EDK II Build Tools	NO	YES	NO	YES	YES	YES

Note: *This document is intended for persons doing EFI development and support for different platforms. It is most likely only of interest in the event that there is a problem with a build, or if a developer needs to perform special customizations of a build for a platform. This document is most likely only of interest in the event that there is a problem with a build, or if a developer needs to perform special customizations of a build for a platform.*

The EDK II build processes, defined in the EDK II Build Specification, use separate steps to create EFI images. The EDK II DSC file is used in conjunction with EDK II Flash Description files (FDF), EDK II DEC, EDK II module INF and EDK component INF files to generate binary PE32/PE32+/Coff files. The EDK II Makefiles, generated by the EDK II parsing tool, contain only enough instructions to build the PE32/PE32+/Coff image files. These makefiles do not contain information on the EFI format for FFS or FV file creation. The Makefiles will support third party compilation tools – Microsoft, Intel and GCC tool chains – and at least one EDK II tool, GenFw. The GenFw tool is used to manipulate the files emitted from the compilation tools.

The EDK II build provides UEFI 2.3 and PI 1.2 (Unified EFI, Inc.) specification-compliant

images. Use of the tools in the EDK Compatibility package can be used for creating earlier versions of EFI 1.10 and/or UEFI 2.0 compliant components. To be clear, EDK II tools do not have the limitation of ECP tools, which can only do EFI 1.10 and UEFI 2.0 images.

1.2 Related Information

The following publications and sources of information may be useful to you or are referred to by this specification:

- *Extensible Firmware Interface Specification*, Intel, 2001, http://developer.intel.com/technology/efi.

Note: *See the glossary sections in the EFI 1.10 Specification for additional required or recommended definitions of terms, abbreviations, documents and specifications:.*

Note: *See the master Framework glossary in the Framework Interoperability and Component Specifications help system for additional required or recommended definitions of terms, abbreviations, documents and specifications:* http://www.intel.com/technology/framework/spec.htm.

- *Unified Extensible Firmware Interface Specification*, Version 2.3, Unified EFI, Inc, 2006, http://www.uefi.org.
- *Platform Initialization Specification*, Version 1.1, Unified EFI, Inc., 2006, http://www.uefi.org.
- *Intel® Platform Innovation Framework for EFI Specifications*, Intel, 2006, http://www.intel.com/technology/framework/.
- http://sourceforge.net/projects/edk2/files/
  - *EDK II Module Writers Guide,* Intel, 2010,.
  - *PCD Infrastructure*, Intel, 2009,.
  - *EDK II User Manual*, Intel, 2010.
  - *EDK II C Coding Standard*, Intel, 2010.
  - *EDK II DSC Specification*, Intel, 2010.
  - *EDK II INF File Specification*, Intel, 2010.
  - *EDK II FDF Specification*, Intel, 2010.
  - *EDK II Build Specification, Intel*, 2010.
  - *VFR Programming Language*, Intel, 2010.

1.3 Terms

The following terms are used throughout this document to describe varying aspects of input localization:

BDS

Framework Boot Device Selection phase.

BNF

BNF is an acronym for "Backus Naur Form." John Backus and Peter Naur introduced for the first time a formal notation to describe the syntax of a given language.

Component

An executable image. Components defined in this specification support one of the defined module types.

DXE

Framework Driver Execution Environment phase.

DXE SAL

A special class of DXE module that produces SAL Runtime Services. DXE SAL modules differ from DXE Runtime modules in that the DXE Runtime modules support Virtual mode OS calls at OS runtime and DXE SAL modules support intermixing Virtual or Physical mode OS calls.

DXE SMM

A special class of DXE module that is loaded into the System Management Mode memory.

DXE Runtime

Special class of DXE module that provides Runtime Services

EBNF

Extended "Backus-Naur Form" meta-syntax notation with the following additional constructs: square brackets "[…]" surround optional items, suffix "*" for a sequence of zero or more of an item, suffix "+" for one or more of an item, suffix "?" for zero or one of an item, curly braces "{…}" enclosing a list of alternatives and super/subscripts indicating between n and m occurrences.

EFI

Generic term that refers to one of the versions of the EFI specification: EFI 1.02, EFI 1.10, UEFI 2.0, UEFI 2.1, UEFI 2.2 or UEFI 2.3.

EFI 1.10 Specification

Intel Corporation published the Extensible Firmware Interface Specification. Intel has donated the EFI specification to the Unified EFI Forum, and the UEFI now owns future updates of the EFI specification.

Foundation

The set of code and interfaces that glue implementations of EFI together.

Framework

Intel® Platform Innovation Framework for EFI consists of the Foundation, plus other modular components that characterize the portability surface for modular components designed to work on any implementation of the Tiano architecture.

GUID

Globally Unique Identifier. A 128-bit value used to name entities uniquely. A unique GUID can be generated by an individual without the help of a centralized authority. This allows the generation of names that will never conflict, even among multiple, unrelated parties. GUID values can be registry format (8-4-4-4-12) or C data structure format.

HII

Human Interface Infrastructure. This generally refers to the database that contains string, font, and IFR information along with other pieces that use one of the database components.

IFR

Internal Forms Representation. This is the binary encoding that is used for the representation of user interface pages.

Library Class

A library class defines the API or interface set for a library. The consumer of the library is coded to the library class definition. Library classes are defined via a library class .h file that is published by a package.

Library Instance

An implementation of one or more library classes.

Module

A module is either an executable image or a library instance. For a list of module types supported by this package, see module type.

Module Type

All libraries and components belong to one of the following module types: BASE, SEC, PEI_CORE, PEIM, DXE_CORE, DXE_DRIVER, DXE_RUNTIME_DRIVER, DXE_SMM_DRIVER, DXE_SAL_DRIVER, UEFI_DRIVER, or UEFI_APPLICATION. These definitions provide a framework that is consistent with a similar set of requirements. A module that is of module type BASE, depends only on headers and libraries provided in the MDE, while a module that is of module type DXE_DRIVER depends on common DXE components. For a definition of the various module types, see Appendix C.

Package

A package is a container. It can hold a collection of files for any given set of modules. Packages may be described as one of the following types of modules:

— source modules, containing all source files and descriptions of a module
— binary modules, containing EFI Sections or a Framework File System and a description file specific to linking and binary editing of features and attributes specified in a Platform Configuration Database (PCD,)
— mixed modules, with both binary and source modules

Multiple modules can be combined into a package, and multiple packages can be combined into a single package.

Protocol

An API named by a GUID.

PCD

Platform Configuration Database.

PEI

Pre-EFI Initialization Phase.

PPI

A PEIM-to-PEIM Interface that is named by a GUID.

Runtime Services

Interfaces that provide access to underlying platform-specific hardware that might be useful during OS runtime, such as time and date services. These services become active during the boot process but also persist after the OS loader terminates boot services.

SAL

System Abstraction Layer. A firmware interface specification used on Intel® Itanium® Processor based systems.

SEC

Security Phase is the code in the Framework that contains the processor reset vector and launches PEI. This phase is separate from PEI because some security schemes require ownership of the reset vector.

SKU

Stock Keeping Unit.

UEFI Application

An application that follows the UEFI specification. The only difference between a UEFI application and a UEFI driver is that an application is unloaded from memory when it exits regardless of return status, while a driver that returns a successful return status is not unloaded when its entry point exits.

UEFI Driver
A driver that follows the UEFI specification.

UEFI Specification Version 2.3
Current version of the EFI specification released by the Unified EFI Forum.

UEFI Platform Initialization Specification 1.2
Current version of the PI specification released by the Unified EFI Forum. This specification builds on the Intel Platform Innovation Framework for EFI specifications and transfers ownership of most of these specifications from Intel to a non-profit, industry trade organization.

Unified EFI Forum
A non-profit collaborative trade organization formed to promote and manage the UEFI standard. For more information, see www.uefi.org.

VFR
Visual Forms Representation.

1.4 Conventions Used in this Document

This document uses typographic and illustrative conventions described below.

1.4.1 Data Structure Descriptions

Intel® processors based on 32 bit Intel® architecture (IA 32) are "little endian" machines. This distinction means that the low-order byte of a multi byte data item in memory is at the lowest address, while the high-order byte is at the highest address.

Processors of the Intel® Itanium® processor family may be configured for both "little endian" and "big endian" operation. All implementations designed to conform to this specification will use "little endian" operation.

In some memory layout descriptions, certain fields are marked reserved. Software must initialize such fields to zero and ignore them when read. On an update operation, software must preserve any reserved field.

The data structures described in this document generally have the following format:

Summary:

A brief description of the data structure.

Prototype:

An EBNF-type declaration for the data structure.

Example:

Sample data structure using the prototype.

1.4.2 Pseudo-Code Conventions

Pseudo code is presented to describe algorithms in a more concise form. None of the algorithms in this document are intended to be compiled directly. The code is presented at a level corresponding to the surrounding text.

In describing variables, a *list* is an unordered collection of homogeneous objects. A *queue* is an ordered list of homogeneous objects. Unless otherwise noted, the ordering is assumed to be FIFO.

Pseudo code is presented in a C-like format, using C conventions where appropriate. The coding style, particularly the indentation style, is used for readability and does not necessarily comply with an implementation of the *UEFI Specification*.

1.4.3 Typographic Conventions

This document uses the typographic and illustrative conventions described below:

Typographic Convention	Typographic convention description
Plain text	The normal text typeface is used for the vast majority of the descriptive text in a specification.
Plain text (blue)	Any plain text that is underlined and in blue indicates an active link to the cross-reference. Click on the word to follow the hyperlink.
Bold	In text, a **Bold** typeface identifies a processor register name. In other instances, a **Bold** typeface can be used as a running head within a paragraph.
Italic	In text, an *Italic* typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.
`BOLD Monospace`	Computer code, example code segments, and all prototype code segments use a `BOLD Monospace` typeface with a dark red color. These code listings normally appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.
`Bold Monospace`	Words in a `Bold Monospace` typeface that is underlined and in blue indicate an active hyperlink to the code definition for that function or type definition. Click on the word to follow the hyperlink.
$(VAR)	This symbol VAR defined by the utility or input files.
`Italic Mono-space`	In code or in text, words in *`Italic Monospace`* indicate placeholder names for variable information that must be supplied (i.e., arguments).

Note: *Due to management and file size considerations, only the first occurrence of the reference on each page is an active link. Subsequent references on the same page will not be actively linked to the definition and will use the standard, non underlined* `BOLD Monospace` *typeface. Find the first instance of the name (in the underlined* `BOLD Monospace` *typeface) on the page and click on the word to jump to the function or type definition.*

The following typographic conventions are used in this document to illustrate the Extended Backus-Naur Form.

[item]	Square brackets denote the enclosed item is optional.
{item}	Curly braces denote a choice or selection item, only one of which may occur on a given line.
<item>	Angle brackets denote a name for an item.
(range-range)	Parenthesis with characters and dash characters denote ranges of values, for example, (a-zA-Z0-9) indicates a single alphanumeric character, while (0-9) indicates a single digit.

"item"	Characters within quotation marks are the exact content of an item, as they must appear in the output text file.
?	The question mark denotes zero or one occurrences of an item.
*	The star character denotes zero or more occurrences of an item.
+	The plus character denotes one or more occurrences of an item.
item$^{\{n\}}$	A superscript number, n, is the number occurrences of the item that must be used. Example: $(0\text{-}9)^8$ indicates that there must be exactly eight digits, so 01234567 is valid, while 1234567 is not valid.
item$^{\{n,\}}$	A superscript number, n, within curly braces followed by a comma "," indicates the minimum number of occurrences of the item, with no maximum number of occurrences.
item$^{\{,n\}}$	A superscript number, n, within curly brackets, preceded by a comma ","indicates a maximum number of occurrences of the item.
item$^{\{n,m\}}$	A super script number, n, followed by a comma "," and a number, m, indicates that the number of occurrences can be from n to m occurrences of the item, inclusive.

2
DSC Overview

This section of the document provides an overview to processing EDK II platform description (DSC) file. Additional chapters describe different sections of the EDK II DSC file in detail.

EDK II parsing tools contain the templates for processing files to create the component binary images from source files. Binary Modules are not included in EDK II DSC files. EDK II DSC files are a list of:

- EDK Component or EDK II Module INF Files
- EDK libraries (only used by EDK Components)
- EDK II Library Class Instance Mappings (only used by EDK II Modules)
- EDK II PCD Entries

Note: *Path and Filename elements within the DSC are case-sensitive in order to support building on UNIX style operating systems.*

2.1 Processing Overview

Each platform DSC file is broken out into sections in a manner similar to the component description (INF) files. However, while the intent of a component's INF file is to define the source files, libraries (or library classes), and definitions relevant to building the component, the function of the platform DSC file is to specify the library instances, components and output formats used to generate binary files that will be processed by other tools to generate an image that is either put into a flash device, made available for recovery booting or updating existing firmware on a platform.

Note: *For users familiar with EDK, the EDK II DSC file is not used to specify how compiled binary images get placed into UEFI/PI compliant binaries. The EDK II Flash Description File (FDF) file specifies how to package the binaries files into UEFI/PI compliant images.*

In general, the parsing utilities read each line from the sections of the platform description (DSC) file, process the component, module, or library's INF file on the line to generate a makefile, and then continue with the next line.

Figure 1 illustrates the process flow, with the dark format indicating the process for building PE/PE32+/Coff files.

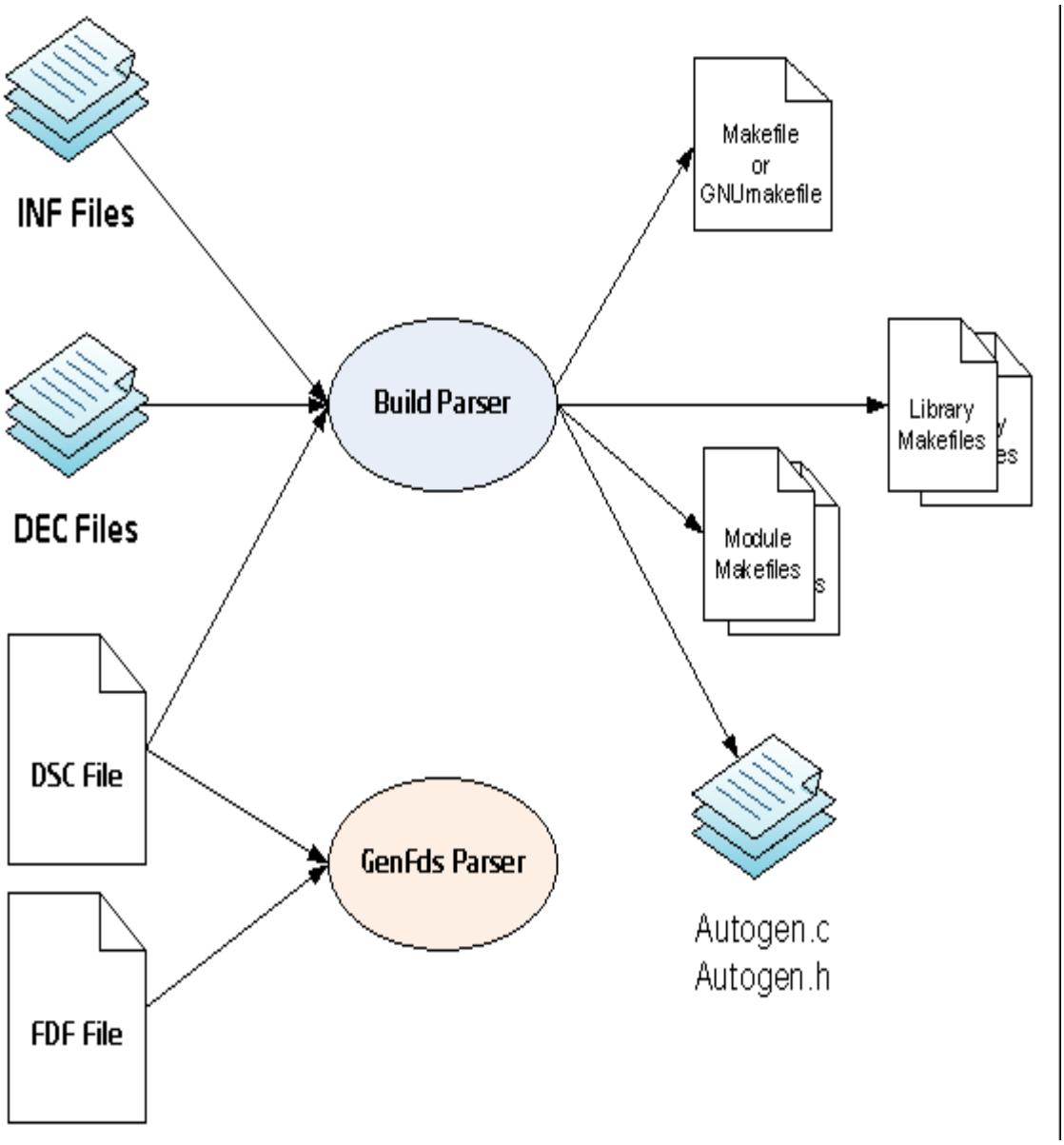

Figure 1. EDK II Parsing Data Flow

Figure 2 illustrates the relationship of the major environment variables, **WORKSPACE**, **EDK_SOURCE, EFI_SOURCE** and **EDK_TOOLS_PATH**. The **EDK_SOURCE** and **EFI_SOURCE** environment variables are only required if using EDK components. The environment variables can point to different locations on a developer's workstation.

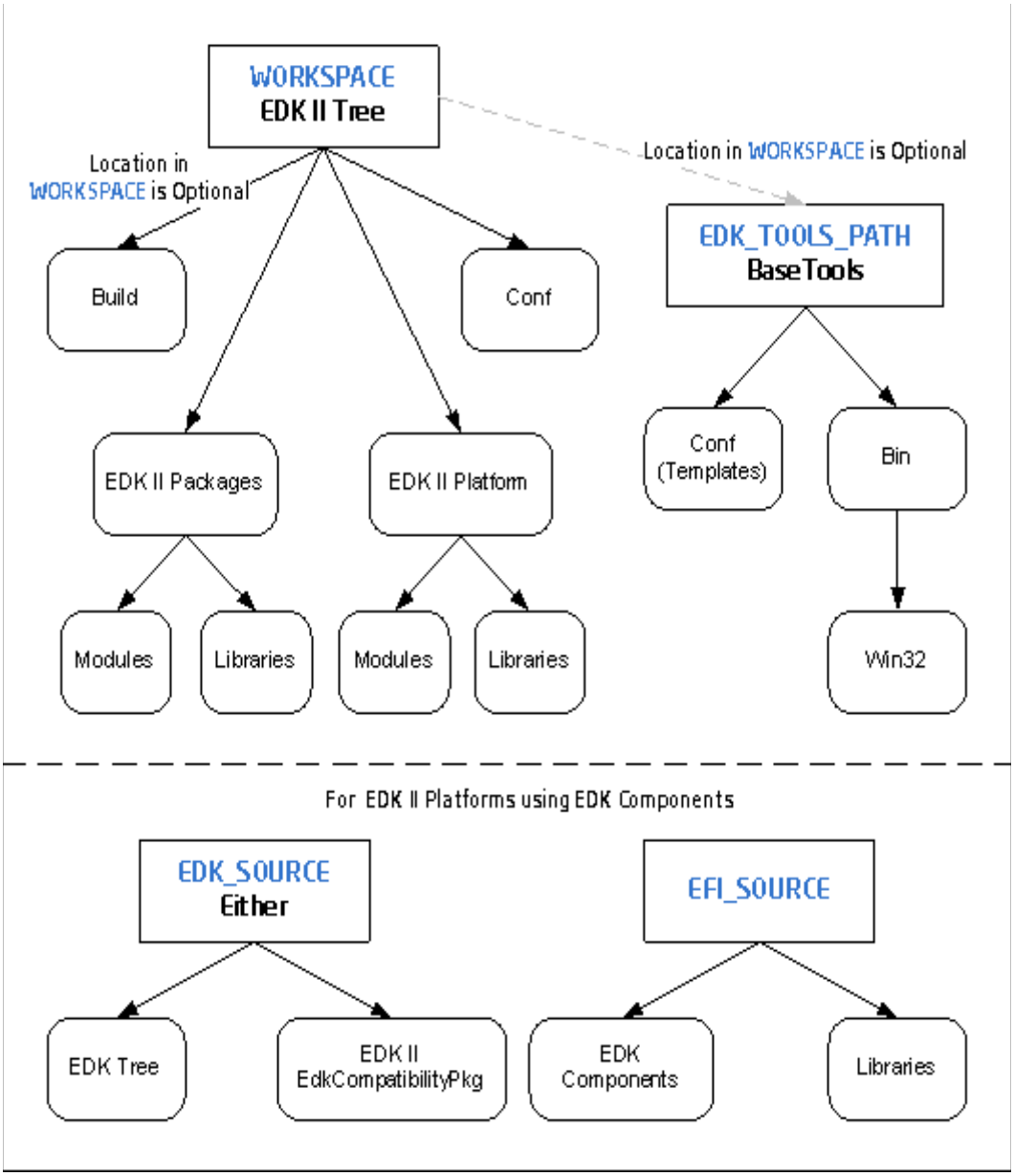

Figure 2. Developer Workstation Layout - EDK II Build

2.2 Build Description File Format

EDK II build description files--DSC, FDF, DEC and INF files, along with other files like build_rules.txt, target.txt and tools_def.txt--contain information used by the parsing utility to

create makefiles that process source files to generate binary (PE32/PE32+/Coff) files. The binary files can be distributed as EDK II binary packages or used to create a platform that is defined in an FDF file, rather than the EDK II DSC file.

2.2.1 Section Entries

This description file consists of sections delineated by section tags enclosed within square [] brackets. Section tag entries are case-insensitive. The different sections and their usage are described below. The text of a given section can be used for multiple section names by separating the section names with a comma. For example:

```
[LibraryClass.X64, libraryclass.ipf]
```

The content below each section heading is processed by the parsing utilities in the order that they occur in the file. The precedence for processing these architecture section tags is from right to left, with sections defining an architecture having a higher precedence than a section which uses common (or no architecture extension) as the architecture.

Note: *Content within a section IS case sensitive. IA32, Ia32 and ia32 within a section are processed as separate items. (Refer to Naming Conventions below for more information on directory and/or file naming.)*

2.2.2 Comments

The hash # character indicates comments in the Platform Description (DSC) file. In line comments terminate the processing of a line. In line comments must be placed at the end of the line, and may not be placed within the section ([,]) tags.

Only `FIX_LOAD_TOP_MEMORY_ADDRESS = 0xF0000000` in the following example is processed by tools; the remainder of the line is ignored:

```
FIX_LOAD_TOP_MEMORY_ADDRESS  = 0xF0000000  # set top memory address
```

Note: *Blank lines and lines that start with the hash # character must be ignored by tools.*

Hash characters appearing within a quoted string are permitted, with the string being processed as a single entity. The following example must handle the quoted string as single element by tools.

```
UI =  "# Copyright 2007, foobar,ltd. All rights reserved."
```

2.2.3 Valid Entries

All entries must appear on a single line, with entries terminated by either a new line, or a comment. Comments must be scripting style, using the hash # character. (See Section 2.2.2.)

2.2.4 Naming Conventions

The EDK II build infrastructure is supported under Microsoft* Windows*, Linux* and MAC OS/X operating systems. As a result of multiple environment support, all directory and file names are case sensitive. Additionally, the use of special characters in directory and file names is restricted to the dash "-", underscore "_" and period "." characters.

Under no circumstances can space characters be used in a directory or file names in EDK II. That said, the build tools must be able to process the tool definitions file: tools_def.txt (describing the location and flags for compiler and user defined tools)

which will contain space characters in paths on Windows* systems. This file is the ONLY file that permits the use of space characters is a directory path.

Additionally, all EDK II directories that are architecturally dependent must use a name with only the first character capitalized. Ia32, Ipf, X64 and Ebc are valid architectural directory names. IA32, IPF and EBC are not acceptable directory names, and may cause build breaks.

Architecture keywords (IA32, IPF, X64 and EBC) are used by build tools and in meta-data files for describing alternate threads for processing of files. These keywords must not be used for describing directory paths. Additionally, directory names with architectural names (Ia32, Ipf, X64 and Ebc) do not automatically cause the build tools or meta-data files to follow these alternate paths. Directories and Architectural Keywords are similar in name only.

2.2.5 !include Statement Processing

The `!include` statement may appear within any section of EDK II DSC file. The included file content must match the content type of the current section definition. Each section will state whether this statement is valid.

Using `!include` statements is NOT recommended. Rather than use the !include statement, platform integrators are encouraged to maintain "common" text files that they would normally include, then use a text editor to cut and paste the content into the appropriate section of the DSC file. This method ensures that a build can be repeatable and reproducible.

2.2.6 Macro Statements

The use of MACRO statements, which assign a value to a variable, are limited in EDK II DSC files to be valid for DSC and FDF files only. The format and usage for the macro statements is:

```
DEFINE MACRO = Path

$(MACRO)/filename.foo
```

Any defined MACRO definitions will be expanded by tools when they encounter the entry in the section. Three environment variables, `$(WORKSPACE)`, `$(EDK_SOURCE)` and `$(EDK_TOOLS_PATH)` are never expanded when data is emitted to Makefiles.

The macro statements are positional, in that only statements following a macro definition are permitted – a macro cannot be used before it has been defined.

Macros may be used in conditional statements located within the DSC file. Conditional MACROs may be defined on a command line of a parsing tool. A macro defined in this manner does not need to have a DEFINE statement to set a value. In the reference build (Nt32Pkg/Nt32Pkg.dsc), macros set on a command line override any macro value defined in the DSC file.

MACROs may also be used as values in PCD statements. See Section 3.9 for more information on PCD statements.

2.2.7 Conditional Directive Statements (!if...)

Conditional directive statements are used by the build tools preprocessor function to include or exclude statements in the DSC file. A limited number of statements are

supported, and nesting of conditionals is also aupported. Statements are prefixed by the exclamation "!" character. Conditional statements may appear anywhere within the DSC file.

Supported statements are:

```
!ifdef, !ifndef, !if, !elseif, !else and !endif
```

The following is an example of conditionals.

```
!ifdef $(FOO)
  !ifndef $(BAR)
     # FOO defined, BAR not defined
  !else
     # FOO defined, BAR is defined
  !endif
!elseif $(BARFOO)
  # FOO is not defined, BARFOO is defined as TRUE
!elseif $(BARFOO) == $(FOOBAR)
  # FOO is not defined, BARFOO is defined as FOOBAR
!else
  # FOO is not defined while BARFOO is either NOT defined or does
  # not equal FOOBAR
!endif
```

2.2.8 [Defines] Section Processing

The defines section of an EDK II DSC file is used to define variable assignments that can be used in later build steps.

Note: *Unlike EDK DSC parsing utilities, later assignments of local symbols override previous assignments.*

This section is required in all EDK II DSC files.

The defines section uses one of the following section definitions:

```
[defines] [defines.IA32] [defines.X64] [defines.IPF] [defines.EBC]
```

The format for entries in this section is:

```
Name = Value
```

All elements of the DSC file's [Defines] section are also valid when parsing the FDF file. The use of the `DEFINE MACRO = Value` statements in this section globally define the MACRO name. The `DEFINE MACRO = Value` statements in other sections are local to the section, and override the global definition for entries in the section that follow the macro definition: `DEFINE MACRO = Value.`

Any defined MACRO will be expanded by tools when they encounter the entry in the section. The following table lists the valid content of this section and whether the item is required.

Table 2. EDK II [Defines] Section Elements

Typical Tag Names	Required/ Optional	Value	Notes
DSC_SPECIFICATION	Required	0x00010005	This entry is required for all EDK II DSC files.
PLATFORM_GUID	Required	Registry Format GUID (8-4-4-4-12)	
PLATFORM_VERSION	Required	Integer or Decimal Number	
PLATFORM_NAME	Required	Single Word	Only alphanumeric, dash and underscore character are permitted
SKUID_IDENTIFIER	Required	Formatted text	This value may be passed on the command line and must match one of the defined names in the [SkuIds] section. If it is passed on the command line, the command line value takes precedence.
SUPPORTED_ARCHITECTURES	Required	List	Pipe (“|”) separated list of architectures that the platform supports
BUILD_TARGETS	Required	List	Pipe (“|”) separated list of build targets (that are defined in the tools_def.txt file)
OUTPUT_DIRECTORY	Optional	Directory	Either a **WORKSPACE** relative or absolute directory location. The default value is: **$(WORKSPACE)/Build/PlatformName**
FLASH_DEFINITION	Optional	Filename	The Filename (FDF) that contains the Flash Part Definition information. The filename may be **WORKSPACE** relative or relative to the location of the DSC file.
BUILD_NUMBER	Optional	Up to four digit number	Set this value in the generated Makefile.
FIX_LOAD_TOP_MEMORY_ADDRESS	Optional	Address	The top memory address - the starting location in memory for all drivers, application loading. When it is not set, or set to 0, the load fixed address feature will be disabled. When it is set to 0xFFFFFFFFFFFFFFFF, enable the feature as fixed offset to TOLM. When it is set to the positive address, enable the feature as fixed address.
BUILD_TYPE	Optional	Set this value in the generated Makefile.	

Typical Tag Names	Required/ Optional	Value	Notes
TIME_STAMP_FILE	Optional	Filename	The timestamp file contains a timestamp that will be used to set the creation timestamp on all created files. If this file is specified, it will be used to adjust the timestamp of created files, if it does not exist at the start of a build, the file will be created, using the current date and time. If this variable is not specified, the time and date of the start of the build are used by the EDK II tools that modify the time/date portion of a PE32/PE32+/Coff header. This is either a `WORKSPACE` relative path and file, or an Absolute Path and filename.

DEFINE	Optional	MACRO = PATH	Either `$(WORKSPACE)` or `$(EDK_SOURCE)` followed by the a path. This statement can be used to make the DSC file more readable, as in:
			`DEFINE MDE = $(WORKSPACE)/MdePkg`
			Then, later,
			`$(MDE)/Library/BaseLib/BaseLib.inf`

RFC_LANGUAGES	Optional	RFC4646 Language code list	A semi-colon ";" separated list of RFC4646 Language codes (EDK II Modules) used during the generation of only a set, rather than all, UNICODE languages during the StrGather autogen phase. The list must be encapsulated in double quotes.
ISO_LANGUAGES	Optional	ISO-639-2 Language code list	A non-separated list of three character ISO 639-2 Language codes (EDK Components) used during the generation of only a set, rather than all, UNICODE languages during the StrGather autogen phase. The list must be encapsulated in double quotes.

Note: *EDK II Modules can have unicode string files that contain RFC4646 language codes. EDK II modules cannot have unicode string files that contain ISO-629-2 language codes. The format of the statement is specific to processing RFC4646 language code lists.*

2.2.9 [BuildOptions] Section

Content in the `[BuildOptions]` section is used to define module specific tool chain flags that should be used as the default flags for a module. These flags are appended to any standard flags that are defined by the build process. They can be applied for any modules or those modules on the specific ARCH or those modules with the specific module style (EDK or EDKII). In order to replace the standard flags that are defined by

the build process, an alternate assignment operator should be used; "==" is used for replacement, while "=" is used to append the flag lines. In addition to flags, other tool attributes may have the item either appended or replaced.

Valid content is within this section is limited to the following description.

Table 3. EDK II [BuildOptions] Section Elements: Optional Tags

Tag	Value	Notes
${FAMILY}:${TARGET}_${TAGNAME}_${ARCH}_${TOOLCODE}_FLAGS	Flags for specific tool codes for this module	Used to specify module specific flags.
${FAMILY}:${TARGET}_${TAGNAME}_${ARCH}_${TOOLCODE}_PATH	The fully qualified path an executable	Used to replace a specific command, such as forcing the ASL to be iasl, instead of asl.
${FAMILY}:${TARGET}_${TAGNAME}_${ARCH}_${TOOLCODE}_DPATH	A fully qualified path	A path that will be added to the system Environment's PATH variable prior to executing a command
${FAMILY}:${TARGET}_${TAGNAME}_${ARCH}_${TOOLCODE}_${ATTRIBUTE}	Attribute specific string	This permits overriding other attributes if required.

Table 4. EDK II [BuildOptions] Variable Descriptions

Variable	Required	Wildcard	Source
FAMILY	NO	No	Conf/tools_def.txt defines FAMILY as one of MSFT, INTEL or GCC. Typically, this field is used to help the build tools determine whether the line is used for Microsoft style Makefiles or the GNU style Makefiles. By not specifying the FAMILY, the tools assume the flags are applicable to all families.
TARGET	YES	Yes = *	Conf/tools_def.txt file defines two values: DEBUG and RELEASE. Developers may define additional targets.
TAGNAME	YES	Yes = *	Conf/tools_def.txt file defines several different tag names – these are defined by developers; the default tag name, MYTOOLS, is provided in the template for tools_def.txt and set in the Conf/target.txt file.
ARCH	YES	Yes = *	Conf/tools_def.txt defines four architectures: IA32, X64, IPF and EBC. This tag must use all capital letters for the tag. Additional Architectures, such as PPC or ARM may be added as support becomes available.
TOOLCODE	YES	NO	The tool code must be one of the defined tool codes in the Conf/tools_def.txt file. The flags defined in this section are appended to flags defined in the tools_def.txt file for individual tools. EXCEPTION: If the INF MODULE_TYPE, defined in the [defines] section is USER_DEFINED, then the flags listed in this section are the only flags used for the TOOLCODE command specified in Conf/tools_def.txt.
ATTRIBUTE	YES	NO	The attribute must be specific to the tool code and must be a valid attribute handled by the build system. For the reference build, a valid rule must be included in the build_rule.txt.

Developers should use extreme caution when specifying items in this section. The EDK II build is designed to support multiple compilers and tool chains, expecting that code is written in ANSI C. If custom tool flags are required by a module, developers must make sure that all consumers of the module are aware of the specific tools and tag names required.

The following examples show the usage of the **[BuildOptions]** section. Note that the lines show use of the "****" line continuation character.

```
[BuildOptions.common]
  MSFT|DEBUG_*_IA32_DLINK_FLAGS = /out:"$(BIN_DIR)\SecMain.exe" \
              /base:0x10000000 /pdb:"$(BIN_DIR)\SecMain.pdb" \
              /LIBPATH:"$(VCINSTALLDIR)\Lib" \
              /LIBPATH:"$(VCINSTALLDIR)\PlatformSdk\Lib" \
              /NOLOGO /SUBSYSTEM:CONSOLE /NODEFAULTLIB /IGNORE:4086 \
              /MAP /OPT:REF /DEBUG /MACHINE:I386
              /LTCG Kernel32.lib MSVCRTD.lib Gdi32.lib User32.lib \
              Winmm.lib
  MSFT|DEBUG_*_IA32_CC_FLAGS = /nologo /W4 /WX /Gy /c /D UNICODE \
              /D EFI32 /Od /DSTRING_ARRAY_NAME=SecMainStrings
              /FI$(DEST_DIR_DEBUG)/AutoGen.h /EHs-c- /GF /Gs8192 \
              /Zi /Gm
```

2.2.10 [SkuIds] Section Processing

The contents of this section are used to define valid `SKUID_IDENTIFIER` names. Since a platform may support different SKUs, and different SKUs may implement different methods for handing platform configuration data (PCD,) the user can define, in this section, tag names to use. Each entry below the section header is of the form:

```
integer | word
```

The following is an example of a `[SkuIds]` section:

```
[SkuIds]
  1|Sku_Two
  22|Sku1
  5|SkuSeven
```

2.2.11 [Libraries] Section Processing

This section specifies all the EDK INF files that must be processed to build the libraries used to build the individual EDK components. This will include all the libraries called out in the individual component INF files. A sample section is listed below. Each line from the libraries section specifies a library component's INF file (relative to `$(EDK_SOURCE)`, or absolute path.)

This section is required for any EDK II DSC file that specifies one or more EDK components. If only EDK II Modules are used, this section should not be specified. If the section is specified, and only EDK II Modules are found, the build and parsing tools will ignore this section. A warning message will be emitted by the parsing tool if and only the parsing tool is executed in a verbose mode.

The `!include` statements may be used within the libraries section.

The file specified after the `!include` statement can only contain a list of EDK Library INF files (with the path to the file). If the line starts with a word, rather than a variable like `$(EDK_SOURCE)` the path is assumed to be relative to `$(EDK_SOURCE)`. Again, only EDK Library INF files are permitted in the file specified in the `!include` statement.

This section will typically use one of the following section definitions:

```
[libraries.common] [libraries.IA32] [libraries.X64] [libraries.IPF]
[libraries.EBC]
```

The formats for entries in this section is:

```
$(EDK_SOURCE)/Path/to/LibraryName.inf
$(CUSTOM_DECOMPRESS_LIB_INF)
```

2.2.12 [LibraryClasses] Section Processing

The [LibraryClasses] section is used to provide a mapping between the library class names used by an EDK II module and the Library Instances that are selected by the platform integrator. Library Classes allow modules to be coded for a library class, and then allow platform integrator then chooses a Library Instance based on a priori knowledge of the instances. Library Instances are classified using the architecture types they have been coded for as well as the supported EDK II module types. As an example, within EDK II, the library class, DebugLib has seven potential instances, only one of which may be linked to a single component. To support a given module type selection, the [LibraryClasses] section header can optionally specify the EDK II module type (following the supported architecture field). This is permitted as some library instances can be used by any or all module types.

This is a an optional section for EDK II DSC files only if there are no EDK II modules used by the DSC file.

2.2.12.1 Generic format for the specifying a section

```
[libraryclasses] [libraryclasses.IA32] [libraryclasses.X64] [libraryclasses.IPF]
[libraryclasses.EBC]
```

2.2.12.1.1 Format for entries in this section

```
LibraryClassName|$(WORKSPACE)/Path/To/LibInstanceName.inf
LibraryClassName1|Path/To/LibInstanceName1.inf
```

Note: *The reserved library class keyword,* `NULL` *is not permitted in any of the* `[libraryclasses]` *sections. The* `NULL` *Library class keyword is only permitted within the* `[Components]` *section's INF file* `<LibraryClass>` *subsection*.

Note: *"LibraryClassName" is a keyword in the first field of the above example format can not be* `NULL`*. The "LibraryClassName" name must be unique to an instance specified in the second field. All INF files that require a LibraryClassName will use this instance when linked to the other libraries or modules.*

The first globally defined library instance, defined in a DSC file, that satisfies a module's requirement for a Library Class, unless specifically overridden by the module in the [Components] section, will be used.

The Library Instances will be selected using the following rules to satisfy a library class for each module listed in the [components] section (in order of highest precedence):

1. `<LibraryClass>` associated with the INF file in the `[Components]` section
2. `[libraryclass.$(Arch).$(MODULE_TYPE), libraryclass.$(Arch).$(MODULE_TYPE)]`
3. `[libraryclass.$(Arch).$(MODULE_TYPE)]`
4. `[libraryclass.common.$(MODULE_TYPE)]`
5. `[libraryclass.$(Arch)]`
6. `[libraryclass.common]` or `[libraryclasses]`

If the Library instance is specified in the context of the INF file (see `[Components]` section), then that library instance will be used. If only a library class is specified in the context of the INF file, then the first matching the `library class | library instance`

following the above precedence rules will be used. If no instance is found for the library class, the build tools should fail with an error similar to the following.

```
ERROR: Library Class [$(LibClassName)] specified by the Module \
   [$(InfFileName)] does not have a Library Class Instance Defined.
```

Build tools should propose at least two fixes:

- Check for spelling of the Library Class Name for the module in the components section, or
- Check the **WORKSPACE** for a library instance that satisfies the Library Class, then add that instance to the DSC file in the correct Library Class section.

The selected library instance is added to the LIBS definition in the output makefile:

```
LIBS = $(LIBS) $(LIB_DIR)/$(LibInstanceName}
```

PCDs that are used by a library instance are resolved using the PCD settings of the driver linking the library instance. Note that if a module's PCD section is used, and multiple modules specify different values for the same PCD setting, there may be multiple instances of the library instance that will be compiled, matching the PCD settings for each module.

2.2.13 PCD Sections Processing

This section is for specifying global (or default) PCD values. The content here is used for generating the AutoGen.c and AutoGen.h files for each of the EDK II modules that are coded for the PcdTokenName if an individual component requires a PCD but does not specify the value.

```
[Pcds(PcdType)] [Pcds(PcdType).IA32] [Pcds(PcdType).X64] [Pcds(PcdType).IPF]
[Pcds(PcdType).EBC]
```

Of the five types of PCDs that have been defined, they fall into one of three categories:

- **FeatureFlag** – which always has a Boolean value.
- **FixedAtBuild** and **PatchableInModule**, which may have a value of one of three data types, Boolean, numeric or pointer.
- **Dynamic** and **DynamicEx**, which may have a value of any one of the three data types. Note that for the dynamic types of PCDs, using an **$(Arch)** extension other than "**common**" in the section header is not valid.

Data Types for PCD values are either Boolean, numeric (**UINT8** - 1 byte, **UINT16** - 2 bytes, **UINT32** - 4 bytes or **UINT64** - 8 bytes) or variable length (**VOID***, which indicates that the value is usually accessed via a pointer). To put a limit on the number of bytes for a variable length value (when the PCD Datum Type is **VOID***) the PCD entry must include the **MaximumDatumSize** parameter. The **MaximumDatumSize** parameter is optional for all other PCD data types.

These are optional sections unless the EDK II modules specified in the **[components]** section use PCDs.

The **[Pcd*]** sections must specify only one line for the PCD item type (within a specified architecture). The item types, **PcdsFeatureFlag**, **PcdsFixedAtBuild**, **PcdsPatchableInModule**, **PcdsDynamic** and **PcdsDynamicEx** are not allowed to be specified within a single [] section.

Incorrect example:

```
[PcdsFixedAtBuild.IA32, PcdsPatchableInModule.IA32, PcdsDynamic.IA32]
```

Unique PCDs are identified using the format to identify the named PCD:

TokenSpaceGuidCName.PcdCName

The content for a PCD in this section is the PCD's Name (PCD Token Space Guid C name, the PCD C name - separated by a period character,) and Default value. Field entries on a line are separated by the pipe | character.

This specification now permits setting different PCD types for a single PCD in different modules. The types here, are PcdsFixedAtBuild, PcdsPatchableInModule, PcdsDynamic and PcdsDyanmicEx. For example, the platform integrator may want to use one module with a given PCD as FixedAtBuild, and have a different module with the same PCD use it as PatchableInModule.

A PCD that is used as **Fixed** for one module, while a different module may want to use the PCD as **Patchable** and a third module might use the PCD as **DynamicEx**. Under normal circumstances, only two of these might be used – **PcdsFixedAtBuild** for modules with well known values for a PCD, then either **PcdsPatchableInModule** or **PcdsDynamicEx** – the first being for testing a module, the second giving the ability for doing individual driver performance tuning "on-the-fly".

Tools must NOT assume that the first method found for a PCD in the PCDs sections will used for all instances of a PCD, but must allow for different modules using a PCD differently, using the **<Pcd*>** statements under the INF file definitions in the **[components]** section.

Precedence for PCDs is as follows:

- If a module entry in the **[components]** section does not specify a PCD value, then the PCD entry is for each INF specified PCD CName will be obtained from the [Pcd*] sections.
- If a module entry in the **[components]** section specifies a PCD value, then that value is used.

Another situation that may result in different PCD types is for a PCD that is declared DYNAMIC in the DEC package definition. A module developer may decide to code their module using on **PcdsFixedAtBuild**, while a different developer may decide to code their module using the same PCD as **PcdsPatchableInModule**.

Library Instances that use PCDs that the module uses must use the same PCD setting as the module using the Library Instance. So if a module uses a PCD as **PcdsFixedAtBuild**, then all library instances that use that PCD must also use the PCD as **PcdsFixedAtBuld**.

Build Tools may detect missing PCD entries (PCD specified in an INF file, but not in the DSC file) and perform one the following:

- Break the build, printing and error message.
- Search the DEC files in the **WORKSPACE**, and use a default value from the DEC file.

PCD Values may be absolute (a number, string, etc.) a MACRO name or an expression. The expression is a C-style expression using C relational, equality and logical numeric and bitwise operators or numeric and bitwise operators that evaluate to a value that matches the PCD's Datum Type (specified in the DEC package declaration file.) Precedence and associativity follow C standards. Using PCDs in expressions is also permitted.

2.2.13.1 [PcdsFeatureFlag] section

The required content for the FeatureFlag PCD is the PCD's C name, an optional PCD Token Space Guid C name, and a Boolean value of either TRUE, FALSE, 1 or 0. These

entries are separated by the pipe "|" character. The following are typical entries, with a supported module type qualifier omitted in these examples:

```
[PcdsFeatureFlag] [PcdsFeatureFlag.common] [PcdsFeatureFlag.IA32]
[PcdsFeatureFlag.X64] [PcdsFeatureFlag.IPF] [PcdsFeatureFlag.EBC]
```

Format of an entry in this section is:

```
PcdTokenSpaceGuidCName.PcdCName|Value
```

Example:

```
gEfiMdeModulePkgTokenSpaceGuid.PcdDxePcdDatabaseTraverseEnabled|1
```

2.2.13.2 [PcdsFixedAtBuild] and [PcdsPatchableInModule] sections

2.2.13.2.1 PcdsFixedAtBuild

The required content for the FixedAtBuild PCD is the PCD's C name, the PCD Token Space Guid C name, and the Value (any one of Boolean, numeric or pointer types). These entries are separated by the pipe "|" character.

If the Datum Type for the PCD is VOID*, then a fourth field that specifies the maximum datum size is required. This is the maximum size allocated by the Platform Integrator. Module developers won't know how much size will be allocated, and just use it. The platform integrator must figure out what the maximum length will be, based on the usage from the modules included.

```
[PcdsFixedAtBuild] [PcdsFixedAtBuild.common] [PcdsFixedAtBuild.IA32]
[PcdsFixedAtBuild.X64] [PcdsFixedAtBuild.IPF] [PcdsFixedAtBuild.EBC]
```

Format of an entry in this section is:

```
PcdTokenSpaceGuidCName.PcdCName|Value[|DatumType[|MaximumDatumSize]]
```

Examples:

```
gEfiMdePkgTokenSpaceGuid.PcdFSBClock | 200000000
gEfiMdeModulePkgTokenSpaceGuidPcdVpdBaseAddress | 0x0
gEfiEdkNt32PkgTokenSpaceGuid.PcdWinNtPhysicalDisk| \
  L"E:RW;245760;512" | VOID* | 32
```

2.2.13.2.2 PcdsPatchableInModule

The required content for the PatchableInModule PCD is the PCD's C name, the PCD Token Space Guid C name, and Value. These entries are separated by the pipe | character. If the Datum Type for the PCD is VOID*, then a fourth field that specifies the maximum datum size is also required.

```
[PcdsPatchableInModule] [PcdsPatchableInModule.IA32] [PcdsPatchableInModule.X64]
[PcdsPatchableInModule.IPF] [PcdsPatchableInModule.EBC]
```

Format of an entry in this section is:

```
PcdCName|PcdTokenSpaceGuidCName|Value[|DatumType[|MaximumDatumSize]]
```

Example:

```
gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000000|UINT32|4
```

2.2.13.3 [PcdsDynamic*] and [PcdsDynamicEx*] sections

2.2.13.3.1 Dynamic*

For Dynamic PCD settings, the section labels must include one of Default, Vpd or Hii with optional architecture and an optional **SKUID_IDENTIFIER** name. The Dynamic entry fields are separated by the pipe | character. If the Datum Type for the PCD is **VOID***,

then a field that specifies the maximum datum size is also required.

For using the standard PCD Get/Set PPI or Protocol.

2.2.13.3.2 Default

```
[PcdsDynamic.$(arch).DEFAULT] [PcdsDynamicDefault.$(arch).$(SkuIdentifier)]
[PcdsDynamicHii.$(arch).$(SkuIdentifier)]
[PcdsDynamicVpd.$(arch).$(SkuIdentifier)]
```

The format for an entry in this section is:

```
PcdTokenSpaceGuidCName.PcdTokenName|Value[|DatumType[|MaximumDatumSize]]
```

Examples:

```
[PcdsDynamicDefault]
gEfiGenericPlatformTokenSpaceGuid.PcdFlashNvStorageVariableBase|0x0
gEfiGenericPlatformTokenSpaceGuid.PcdFlashNvStorageVariableSize|0x0
```

2.2.13.3.3 HII

For using the HII for PCD data, the section name is as follows:

```
[PcdsDynamicHii.$(arch).DEFAULT]
```

Specifying a SKUID_IDENTIFIER name for an Hii Pcd selection is optional, for example:

```
[PcdsDynamicHii.common.Sku1]
```

While the format for content of this section is as follows, note that the backslash character is used here to indicate the continuation of the line:

```
PcdTokenSpaceGuidCName.PcdTokenName|VariableName|VariableGuid \
  | VariableOffset | HiiDefaultValue [|MaximumDatumSize]
```

Examples:

```
[pcdsDynamicHii.common.Sku_Two]
gEfiGmchTokenSpaceGuid.PcdGmchPegPreAllocatedDvmt| 0x0053 0x0065 \
  0x0074 0x0075 0x0070 | gSysconfigGuid | 0x000000A9 | 0x3
```

2.2.13.3.4 VPD

For using the VPD for PCD data, the section name is:

```
[PcdsDynamicVpd.$(arch).DEFAULT]
```

Specifying a SKUID_IDENTIFIER for a VPD PCD selection is optional, for example:

```
[PcdsDynamicVpd.common.Vpd.SkuSeven]
```

While the format for content of this section is as follows:

```
PcdTokenSpaceGuidCName.PcdTokenName|VpdOffset[|MaximumDatumSize]
```

Examples:

```
[PcdsDynamicVpd.IA32.DEFAULT, PcdsDynamicVpd.x64.DEFAULT]
gEfiPhonyTokenSpaceGuidCName.PcdVpdCopyrightLine|0x000000A0
```

2.2.13.3.5 DynamicEx*

Entries for DynamicEx are identical to the Dynamic entries. The DynamicEx entry fields are separated by the pipe | character. If the Datum Type for the PCD is `VOID*`, then MaximumDatumSize field that specifies the maximum datum size is required.

```
[PcdsDynamicEx.$(arch).Default] [pcdsDynamicExDefault.$(arch).$(SkuIdentifier)]
[pcdsDynamicExHii.$(arch).$(SkuIdentifier)]
[pcdsDynamicExVpd.$(arch).$(SkuIdentifier)]
```

The format for an entry in this section is:

```
PcdTokenSpaceGuidCName.PcdTokenName|Value[|DatumType[|MaximumDatumSize]]
```

Examples:

```
[PcdsDynamicEx.common.Default]
PcdFlashNvStorageVariableBase|gEfiGenericPlatformTokenSpaceGuid|0x0
PcdFlashNvStorageVariableSize|gEfiGenericPlatformTokenSpaceGuid|0x0
```

2.2.13.3.6 HII

For using the HII for PCD data, the section name is as follows:

```
[PcdsDynamicExHii.$(arch).$(SKUID_IDENTIFIER)]
```

Specifying a SKUID# for an HII PCD selection is optional, for example:

```
[PcdsDynamicExHii.common.Sku1]
```

While the format for content of this section is as follows, note that the backslash character is used here to indicate the continuation of the line:

```
PcdTokenSpaceGuidCName.PcdTokenName|VariableName| \
VariableGuid | VariableOffset | HiiDefaultValue [|MaximumDatumSize]
```

Examples:

```
[PcdsDynamicExHii.IA32.Sku_Two]
gEfiGmchTokenSpaceGuid.PcdGmchPegPreAllocatedDvmt| \
0x0053 0x0065 0x0074 0x0075 0x0070 | gSysconfigGuid | 0x000000A9 | 0x11
```

2.2.13.3.7 VPD

For using the VPD for PCD data, the section name is:

```
[PcdsDynamicExVpd.$(arch).$(SKUID_IDENTIFIER)]
```

Specifying a SKUID# for a VPD PCD selection is optional, for example:

```
[PcdsDynamicExVpd.common.SkuTwo]
```

The format for content of this section is as follows:

```
PcdTokenSpaceGuidCName.PcdTokenName|VpdOffset[|MaximumDatumSize]
```

Examples:

```
[PcdsDynamicExVpd.common.DEFAULT]
gEfiPhonyTokenSpaceGuidCName.PcdVpdCopyrightLine | 0x000000A0
```

2.2.14 [Components] Section Processing

One or more `[Components]` sections contain lists of EDK components and EDK II Modules. The format for specifying the INF file for EDK II modules incorporates new scoping capabilities.

This section uses one or more of the following section definitions:

```
[Components] [components.IA32] [components.X64] [components.IPF]
[components.EBC]
```

EDK components are specified using a fully qualified path to the EDK INF file.

```
$(EDK_SOURCE)/Path/and/Filename.inf
```

Because EDK II modules have different requirements than EDK I components, specifying the INF filename in the extended DSC file may require more than just the INF filename and options. A scoping structure, that binds library class (with an optional override instance,) PCD settings (also overriding the values specified in the `[pcd{PcdType}]` section) and build options to an EDK II module may be required. This scoping structure, containing sub-elements, is enclosed within curly braces: "{}". The opening curly brace, "{", must appear at the end of the inf filename line, before any comments.

These are needed only if specifying a non-default library class (one specified in the `[LibraryClasses]` section), or if changing a PCD value from defaults specified in the

`[pcd{PcdType)]` section, or ir overriding tool flags set either in the `[Defines]` section of the DSC file, or if an entry in the tools_def.txt file.

Note: Section 3.10 *defines the sub-element content of an INF file specified in a component section. This code paragraph section shows the format of a "scoping structure".*

```
$(WORKSPACE)/Path/and/Filename.inf {
# Sub-elements - See EDK II INF file statement structure
}
```

There are three valid, optional sub-elements for EDK II modules. These sub-element are enclosed within angle brackets: `<LibraryClass>`, `<Pcd*>` and `<BuildOptions>`.

An INF file line may also have one argument, EXEC = Filename, that specifies an executable file that takes the INF filename as a parameter. The Filename must be executable, and must take the INF filename. No other arguments are permitted to the Filename.

The parsing tools will call the executable specified by the Filename, as follows:

```
$Filename $(WORKSPACE)/Path/and/Filename.inf
```

If the EXEC argument is included on the component or module INF line, EDK II Tools will ignore processing of the module. Additionally, EDK II build tools will not perform any dependency checking for files listed in the INF file nor on the output object file. EXEC example follows:

```
$(WORKSPACE)/Path/and/Filename.inf EXEC = exe2bin.exe
```

The structure for specifying an EDK II INF filename is as follows:

```
$(EDK_SOURCE)/Path/and/ComponentName.inf [options] {
  # Library Class listing - Required for EDK II components. One or more
  # Library mapping lines are permitted.
  <LibraryClass>
    LibraryClassName|$(WORKSPACE)/Path/and/LibraryInstanceName.inf
  # Followed by zero (no pcds are used by the module) or more of the
  # following individual component override statements.
  # Pcd settings are applicable to both Library Instances that
  # are linked to the component and the component itself.
  # If a PCD entry is not specified, however the INF files for either
  # the library classes that are linked to the component, or the
  # component's INF file require Pcd settings, a globally defined
  # value for the Pcd ([Pcd{PcdType}] section) will be used.  If
  # neither of these is specifed, build tools processing this file
  # should fail with an error message (indicating the missing PCD
  # entry.)
  # NOTE: MaximumDataSize is required for VOID* datum type PCDs, and
  # should NOT be used for either the boolean or numeric datum types.
  # Additionally, for the PcdsDynamic and PcdsDynamicEX sub-elements,
  # the modifiers of DEFAULT, VPD HII and SKUID are permitted.
  <PcdsFixedAtBuild>
    TokenSpaceGuidCName.PcdTokenName|Data[|MaximumDataSize]
  <PcdsFeatureFlag>
    TokenSpaceGuidCName.PcdTokenName|{TRUE}{FALSE}
  <PcdsPatchableInModule>
    TokenSpaceGuidCName.PcdTokenName|Data[|MaximumDataSize]
  <PcdsDynamic>
    TokenSpaceGuidCName.PcdTokenName|Data
  <PcdsDynamicEx>
    PcdTokenName
   # BuildOption Format is identical to the format used for the
   #tools_def.txt file. Options specified in this section take
   # precedence over the platform build options specified in the
   #[Defines] section.  (The [Defines] section options take precedence
   #over options specified in the tools_def.txt file)
  <BuildOptions>
    TARGET_TOOLCHAIN_ARCH_COMMANDTYPE_FLAGS = string of flags
}
```

If the values for the `PcdTokenName` are specified in the global `[pcd{PcdType}]` section and no values are specified in the INF filename section, then the global values will be used. It should be noted that only `library class | library instance` definitions are required. Adding just the library class name without the library instances is for informational purposes only, and they will not be processed.

Example:

```
$(EDK_SOURCE)/MdeModulePkg/Universal/Disk/DiskIo/Dxe/DiskIo.inf { #EDK II Module
  <LibraryClass>
DebugLib|$(WORKSPACE)/MdePkg/Library/PeiDxeDebugLibReportStatusCode/
PeiDxeDebugLibReportStatusCode.inf
BaseMemoryLib|$(WORKSPACE)/MdePkg/Library/DxeMemoryLib/DxeMemoryLib.inf
MemoryAllocationLib|$(WORKSPACE)/MdePkg/Library/DxeMemoryAllocationLib/
DxeMemoryAllocationLib.inf
}
```

2.2.15 [UserExtensions] Section

Users may develop custom tools that use the `[UserExtensions]` sections.The EDK II `[UserExtensions]` sections allow for extending the DSC file with custom processing of component images. The format for a user extension section specifier is "`[UserExtensions.$(UserID).$(Identifier)]`".

The EDK II build tools available on TianoCore.org do not use this section. The reference tools ignore all content within a `[UserExtensions]` section.

The following is an example of a `[UserExtensions]` section:
`[userextensions.Edk2AcpiTable."POST_PROCESSING"]`.

3
EDK II DSC Format Specification

3.1 Introduction

This section of the document describes the DSC sections using an Extended Backus-Naur Form. It should be noted that the DSC file must define all libraries, components and/or modules that will be processed by third party compiler tool chains, such as the GNU C compiler (gcc). Binary Only modules do not need to be listed in this file, as they can be specified in the FDF file. There may also be modules listed in this section that are not required in the FDF file. When a module listed here is excluded from FDF file, then a UEFI-compliant binary will be generated, but the binary will not be put into any firmware volume.

Note: *Path and Filename elements within the DSC are case-sensitive in order to support building on UNIX style operating systems.*

3.2 Platform DSC Definition

Summary

Platform EDK II DSC description

```
<EDK_II_DSC>          ::=  [<Header>]
                           [<IncludeStatement>]
                           [<Defines>]
                           [<SkuIds]>
                           [<Libraries>]
                           [<LibraryClasses>]
                           [<Pcds>]
                           [<Components>]
                           [<BuildOptions>]
                           [<UserExtensions>]
```

Note: *Assignments set as command-line arguments to the parsing tools take precedence over all assignments defined in the DSC file.*

The general rules for content in EDK II DSC files follows.

To append comment information to any item, the comment must start with the hash "#" character and the first occurrence of the comment character terminates processing of a line. The comment terminates with the end of line character. Comments may appear anywhere within the file.

A section terminates with either another section definition or the end of the file.

Field separators for single lines that contain more than one field is the pipe "|" character. This character was selected to reduce the possibility of having the field separator character appear in a string, such as a filename or text string. Whitespace charcacters are permitted between the field separator and the content. The only notable exception is the PcdName which is a combination of the PcdTokenSpaceGuidCName and the PcdCName that are separated by the period ".".

character. This notation for a PCD name was used to uniquely identify the PCD. Whitespace characters are not permitted between the PcdTokenSpaceGuidCName and the dot, nor are they permitted between the dot and the PcdCName.

Note: *The backslash character "\" may be used to extend a line.*

Unless otherwise noted, all file names and paths are relative the system environment variable, **WORKSPACE**. A directory name that starts with a word is assumed by the build tools to be located in the **WORKSPACE** directory.

It should be noted that conditional statements may be used anywhere within the DSC file, with the ability to group any item within a section as well as entire sections.

3.2.1 Common Definitions

Summary

The following are common definitions used by multiple section types.

Prototype

```
<Word>                ::=  Alphanumeric characters (either text or
                           UNICODE) with optional dash "-" and/or
                           underscore "_" characters. Whitespace characters
                           are not permitted.

<FileSep>             ::=  "/"

<FieldSeparator>      ::=  "|"

<Extension>           ::=  One or more alphanumeric characters.

<CName>               ::=  A valid C variable Name

<CFlags>              ::=  <String>

<String>              ::=  {<UnicodeString>} {<AsciiString>}

<BareString>          ::=  <Word> [<Whitepace> <NonWhitesapce>]*

<QuotedString>        ::=  {"'" <BareString> "'"} {"""" <BareString> """"}

<AsciiString>         ::=  {<BareString> } {<QuotedString>}

<Comment>             ::=  "#" [<AsciiString>]

<UnicodeString>       ::=  "L" <QuotedString>
```

```
<HexDigit>            ::=  (a-fA-F0-9)

<HexNumber>           ::=  "0x" [<HexDigit>]{1,}

<RegistryFormatGUID>  ::=  <Hex8> "-" <Hex4> "-" <Hex4> "-" <Hex4> "-" <Hex12>

<Hex8>                ::=  "0x" <HexDigit>{8}

<Hex4>                ::=  "0x" <HexDigit>{4}

<Hex12>               ::=  "0x" <HexDigit>{12}

<CFormatGUID>         ::=  "{ 0x" <HexDigit>{8} ", 0x" <HexDigit>{4}
                           ", 0x" <HexDigit>{4} ", { 0x" <HexDigit>{2}
                           ", 0x" <HexDigit>{2} ", 0x" <HexDigit>{2}
                           ", 0x" <HexDigit>{2} ", 0x" <HexDigit>{2}
                           ", 0x" <HexDigit>{2} ", 0x" <HexDigit>{2}
                           ", 0x" <HexDigit>{2} " }}"

<Integer>             ::=  [0-9]+

<DecimalNumber>       ::=  [{"+"} {"-"}] <Number>

<Number>              ::=  <Integer> ["." <Integer>]

<TrueFalse>           ::=  {"TRUE"} {"FALSE"}

<TRUE>                ::=  {"TRUE"} {"true"}

<FALSE>               ::=  {"FALSE"} {"false"}

<ZeroOne>             ::=  {["0x"] "1"} {["0x"] "0"}

<BoolType>            ::=  {<TRUE>} {<FALSE>} {<ZeroOne>}

<WhiteSpace>          ::=  {<Tab>} {<Space>}

<NonWhiteSpace>        ::=  {<Word>} {<SpecialCharacters>} {<Punctuation>}

<SpecialCharacters>    ::=  One of the following: @#%^&*()+=|}{"></[]\`~'"

<Punctuation>          ::=  One of ".", ",", "!", ":", ";" or "?"

<EOL>                  ::=  End Of Line: "\n" character
```

Note: *Comments may appear anywhere within a DSC file, provided they follow the rules that a comment may not be enclosed within Section headers, and that in line comments must appear at the end of a statement.*

3.2.2 MACRO Statements

Macro statements are characterize by a DEFINE token or may be defined on a

command line of a parsing tool. Command-line definitions take precedence over macro statements specified within the DSC file. Definition statements appearing in a section of the file, and are scoped to the section they are defined in. If the Macro statement is within the **[Defines]** section, then the Macro is common to the entire file, with local definitions taking precedence (if the same MACRO name is used in subsequent sections, then the MACRO value is local to only that section.) Macros defined in the **[Defines]** section may also be used when processing other files specified by the DSC file or by files required by the build system.

Note that a macro named MDEPKG_NDEBUG is reserved for size reduction purposes. The user should not use this keyword to define new macro.

Macro statements may not be referenced before they are defined.

Prototype

```
<MacroDefinition>      ::=  "DEFINE" <MACRO> "=" [{<PATH>} {<Value>}] <EOL>

<MACRO>                ::=  [A-Z]{1,}[_A-Z0-9]{,}

<PATH>                 ::=  [<AbsolutePath>] <RelativePath>

<AbsolutePath>         ::=  {"$(WORKSPACE)/"} {"$(EDK_SOURCE)/"} {<SysEnv>}
                            {"$(EFI_SOURCE)/"} {[a-zA-Z]":\"} {"\"} {"/"}

<SysEnv>               ::=  "$(" <Word> ")/"

<RelativePath>         ::=  <Word> "/" [<Word> "/"]*

<Value>                ::=  {<Number>} {<QuotedString>} {<PcdName>}
                            {<BoolExpression>} {<Expression>}

<Number>               ::=  {<Integer>} {<Hex>}

<Integer>              ::=  (0-9){1,}

<Hex>                  ::=  "0x" (a-fA-F0-9){1,}

<PcdName>              ::=  <TokenSpaceGuidCName> "." <TokenCName>

<TokenCName>           ::=  <CName>

<TokenSpaceGuidCName>  ::=  <CName>

<QuotedString>         ::=  ["L"] {<SingleQuotes>} {<DoubleQuotes>}

<SingleQuotes>         ::=  "'" <String> "'"

<DoubleQuotes>         ::=  '"' <String> '"'

<LogicalOp>            ::=  {<AndOp>} {<OrOp>} {"XOR"} [<NotOp>]

<AndOp>                ::=  {"AND"} {"&&"}

<OrOp>                 ::=  {"OR"} {"||"}

<NotOp>                ::=  {"NOT"} {"!"}
```

Note: *As noted in the EBNF above, the logical operators, "NOT", "AND" and "OR" may be used in addition to the C logical operators. Additionally, a special exclusive or operator, "XOR" has been introduced as a logical, versus bitwise, operator.*

Parameters

BoolExpression

C-style expression using C relational, equality and logical numeric and bitwise operators that evaluate to either TRUE (1) or FALSE (0). Values other than zero or one are invalid. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For

both macro names and PCDs, the element must be previously defined before it can be used.

Expression

C-style expression using C relational, equality and logical numeric and bitwise operators and/or arithmetic and bitwise operators that evaluate to a value (for PCDs, the value must match the Datum Type of the PCD). Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

Examples:

```
DEFINE UEFI =
DEFINE GEN_SKU = $(WORKSPACE)/MyPlatformPkg/GenPei
DEFINE SKU1    = $(WORKSPACE)/MyPlatformPkg/Sku1/Pei
```

3.2.3 Conditional Directive Statements

The conditional statements may appear anywhere within the file. Conditional directive sections can be nested. All conditional directives use macro or Feature Flag PCD values, which must evaluate to either 'True' or 'False.'

Directives must be the only statement on a line.

String evaluations are not permitted.

The expression immediately following the '!if' statement controls whether the content after the line is processed or not. TRUE is any non-zero and/or non-null value other than zero.

Each '!if' within the source code must be matched with a closing '!endif.'

Zero or more #elif statements are permitted, and may be nested, however only one #else statement is permitted.

Directives must encapsulate either entire sections or elements within a single section, such that they do not break the integrity of the section definitions.

Directives are in-fix expressions that are evaluated left to right; content within parenthesis is evaluated before the outer statements are evaluated. Use of parenthesis is recommended to remove ambiguity.

The values of the Feature Flag PCDs used in the conditional statements must be set in the [PcdsFeatureFlag] section(s) of this DSC file. Since this section may appear after a Feature Flag PCD is used in a conditional directive statement, the reference build tools must perform two passes on this file, first to get the values of the Feature Flags which are used for the conditional directives (these feature flags should not be located within a conditional directive,) and during the second pass, to evaluate the conditional statements for inclusion in the build. If the value of a Feature Flag PCD cannot be determined during the first pass, the build should break.

Prototype

```
<Conditional>          ::=  <IfStatement> <EOL>
                            [<Conditional>]{0,}
                            <Statements>{1,}
                            [<ElseIfConditional>]{0,}
                            [<Conditional>]{0,}
                            [<ElseConditional>]{0,1}
                            [<Conditional>]{0,}
                            "!endif" <EOL>

<IfStatement>          ::=  {"!if" <expression>}
                            {"!ifdef" <MACRO>}
                            {"!ifndef" <MACRO>}

<Statements>           ::=  {<Sections>} {<SubSections>}
                            {<SectionStatements>}

<ElseIfConditional>    ::=  "!elseif" <expression> <EOL>
                            <Statements>{1,}

<ElseConditional>      ::=  "!else" <EOL>
                            <Statements>{1,}

<expression>           ::=  {<FeatureFlagExpress>} {<LogicalExpression>}

<FeatureFlagExpress>   ::=  ["NOT"] <PcdFeatureFlag>
                            [<LogicalOp> <PcdFeatureFlag>]*

<PcdFeatureFlag>       ::=  [<PcdTokenSpaceGuidCName> "."] <PcdCName>

<LogicalExpression>    ::=  ["("] <VariableOrTest>
                            [<LogicalOp> <VariableOrTest>]* [")"]

<VariableOrTest>       ::=  {<Variable>} {<AritmeticTest>}
                            {<FeatureFlagExpress>}

<Variable>             ::=  {<FixedPcdName>} {<PatchPcdName>} {"$(" <MACRO> ")"}

<FixedPcdName>         ::=  [<PcdTokenSpaceGuidCName> "."] <PcdCName>

<PatchPcdName>         ::=  [<PcdTokenSpaceGuidCName> "."] <PcdCName>

<MACRO>                ::=  (a-zA-Z)(a-zA-Z0-9_){0,}

<ArithmeticTest>       ::=  ["("] <Variable> <ArithmeticOp>
                           <VariableOrArgument> [")"]

<VariableOrArgument>   ::=  {<Varaible>} {<Argument>}

<Arugment>             ::=  {"0x"(a-fA-F0-9){1,}} {(0-9){1,}}

<LogicalOp>            ::=  {"AND"} {"OR"} {"XOR"} ["NOT"]
```

```
<AritmeticOp>        ::=  {"=="} {"!="} {"<"} {">"} {"<="} {">="}
```

Parameters

Sections

One or more full or partial sections may be within conditional statements. A section starts with the left bracket "[" and terminates with either another section or the end of the file.

Subsections

One or more full or partial subsections may be within conditional statements. A subsection starts with the left arrow "<" and terminates with either another subsection, another section or the end of the file.

SectionStatements

One or more individual elements may be within the scope of the conditional statements.

Variable

Variable values must be numeric or boolean in nature. Tools will cast all variable values to unsigned long long (UINTN) when performing evaluations on values that are of different datum types.

PcdFeatureFlag

The Feature PCD is a boolean PCD that is set to either True (1) or False (0). The PCD datum type for a Feature PCD is always BOOLEAN.

FixedPcdName

A Fixed PCD will have a set value at build time, and that can not be modified in the binary image, nor modified at runtime. For directives, the PCD datum type is limited to UINT8, UINT16, UINT32, UINT64, UINTN and BOOLEAN. Using a PCD that has a datum type of VOID* is prohibited in directives.

PatchPcdName

A PCD has a set value at build time, and that can be modified in the binary image. For directives, the PCD datum type is limited to UINT8, UINT16, UINT32, UINT64, UINTN and BOOLEAN. Using a PCD that has a datum type of VOID* is prohibited in directives.

Example:

```
!if $(IPF_VERSION) == "1"
  [VTF.IPF.MyBsf]
    !ifdef $(IA32RESET)
      # IPF_VERSION is 1 and IA32RESET defined
      IA32_RST_BIN = IA32_RST.BIN
    !endif
    COMP_NAME = PAL_A
    COMP_LOC  = F
    COMP_TYPE = 0xF
    COMP_VER  = 7.01
    COMP_CS   = 1
    !if $(PROCESSOR_NAME) == "M1"
      COMP_BIN  = M1PalCode/PAL_A_M1.BIN
      COMP_SYM  = M1PalCode/PAL_A_M1.SYM
    !elseif PROCESSOR_NAME == "M2"
      COMP_BIN  = M2PalCode/PAL_A_M2.BIN
      COMP_SYM  = M2PalCode/PAL_A_M2.SYM
    !else
      COMP_BIN  = GenPal/PAL_A_GEN.bin
      COMP_SYM  = GenPal/PAL_A_GEN.sym
    !endif
    COMP_SIZE = -
!elseif $(IPF_VERSION) == "2"
  [VTF.IPF.MyBsf]
    !ifdef $(IA32RESET)
      # IPF_VERSION is 2 and IA32RESET defined
      IA32_RST_BIN = IA32_RST.BIN
    !endif
    COMP_NAME = PAL_A
    COMP_LOC  = F
    COMP_TYPE = 0xF
    COMP_VER  = 7.01
    COMP_CS   = 1
    COMP_BIN  = GenPal/PAL_A_GEN.bin
    COMP_SYM  = GenPal/PAL_A_GEN.sym
    COMP_SIZE = -

    COMP_NAME = PAL_B
    COMP_LOC  = F
    COMP_TYPE = 0x01
    COMP_VER  = -
    COMP_CS   = 1
    COMP_BIN  = GenPal/PAL_B_GEN.bin
    COMP_SYM  = GenPal/PAL_B_GEN.sym
    COMP_SIZE = -
!else
  [VTF.X64.MyVtf]
      IA32_RST_BIN = IA32_RST.BIN
!endif
```

3.2.4 !include Statements

Summary

This section defines the `!include` statement in EDK II Platform (DSC) files. This statement is used to include, at the statement's line, a file which is processed at that point, as though the text of the included file was actually in the DSC file. The included file's content must match the content of the section that the `!include` statement resides, or it may contain completely new sections. If the included file starts with a section header, then the section being processed in the Platform DSC file is considered to have been terminated.

The `!include` file cannot contain additional `!include` statements.

Statements in the include file are permitted to override previous definitions as well as to define new entries.

Use of this statement is optional.

Prototype

```
<IncludeStatement>      ::=  "!include" <Filename> <EOL>

<Filename>              ::=  [<PATH>] <Word> ["." <Word>]

<PATH>                  ::=  [<SysEnvVariable>] [<Path>]*

<Path>                  ::=  <Word> "/"

<SysEnvVariable>        ::=  {"$(EDK_SOURCE)/"} {"$(WORKSPACE)/"} {<MACRO>}

<MACRO>                 ::=  "$(" <WORD> ")"
```

Example (EDK II DSC)

```
!include "$(WORKSPACE)/MyPlatformPkg/feature_pcds.mak"
```

Note: *The extension used in the example, `mak`, is just a three character extension, and would not processed by make or nmake commands. It might just as well have been "foo".*

3.3 Header Section

This is an optional section.

The Copyright and License notices for the DSC file are in the comments that start the file. The format for the comment section is:

```
## @file
# Abstract
#
# Description
#
# Copyright
#
# License
#
##
```

Developers will create this section manually (or with the help of usage enhancement tools). It is recommended that the developer maintain a text file that contains the Copyright and License information, which can then be copied into a new DSC file.

Prototype

```
<Header>            ::=  "## @file" <EOL>
                         [<Abstract>]
                         [<Description>]
                         [<Copyright>]
                         [<License>]
                         "##" <EOL>

<FileName>          ::=  <Word> "." <Extension>

<Extension>         ::=  [a-zA-Z][a-zA-Z0-9]*

<Abstract>          ::=  "#" <Sentence> <EOL>
                         "#" <EOL>

<Description>       ::=  ["#" <Sentence> <EOL>]+
                         "#" <EOL>

<Copyright>         ::=  "#" "Copyright (c) " <Date> "," <Company> <EOL>
                         "#" <EOL>

<License>           ::=  ["#" <LicenseSentence> <EOL>]+
                         "#" <EOL>
```

Example:

```
## @file
# EFI/Framework Emulation Platform
#
# The Emulation Platform can be used to debug individual modules, prior
# to creating a real platform.
#
# Copyright (c) 2006 - 2007, Intel Corporation.  All rights reserved.
#
# This program and the accompanying materials are licensed and made
# available under the terms and conditions of the BSD License which
# accompanies this distribution.
# The full text of the license may be found at:

#   http://opensource.org/licenses/bsd-license.php
#
# THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS,
# WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR
# IMPLIED.
#
##
```

3.4 [Defines] Section

Summary

Defines the content of the **[Defines]** section used in EDK II DSC files. This section is created by the developer and is an input to the EDK II parsing utilities. Elements may appear in any order. Some of the entries in this section are emitted to the output makefiles.

This is a required section for EDK II DSC files.

Entries in this section are global to the both the DSC and FDF files and may be used as MACRO statement using the form **$(MACRO_NAME)**.

The value of **OUTPUT_DIRECTORY** can be referenced in other meta-data and makefiles as **$(OUTPUT_DIRECTORY)**. Likewise, **BUILD_NUMBER** can be referenced in later on in this file (or may be used in files specified in the DSC file) as **$(BUILD_NUMBER)**.

Some entries are used by build tools in conjunction with other meta-data files and command-line options for context sensitive processing. The **BUILD_TARGETS** value describes the possible build targets that can be built, with tools_def.txt, target.txt and command-line options limiting the context of the current build process. For example, if **BUILD_TARGETS** is set to **DEBUG|RELEASE**, and the two tool configuration files, target.txt and tools_def.txt permit building both of these targets, and no target is specified on the command-line of the build command, the tool will perform two builds of the platform, one DEBUG followed by a build for **RELEASE**. The two entries that are scoped in this manner are **SUPPORTED_ARCHITECTURES** and **BUILD_TARGETS**. The these items may be referenced as **$(TARGET)** and **$(ARCH**) in other meta-data files.

Sections **[Defines.$(Architecture)]** and **[Defines]** are processed by the EDK II parser tools in the order they appear in the document.

Note: *Assignments of variables in other sections take precedence over previous assignments.*

New EDK II DSC files must include the statement: **DSC_SPECIFCIATION = 0x00010015** in this section.

Individual items must appear on a single line, they may not span multiple lines.

Of special note is the **FLASH_DEFINITION** file name. Unlike other file names in the document, if the value does not include a full path, the file name is relative to the location of the DSC file and NOT relative to the **WORKSPACE**. If the **FLASH_DEFINITION** entry is not specified, the build tools assume that the target output is one or more UEFI_APPLICATION modules.

The **OUTPUT_DIRECTORY** value must be either a full path to the output directory, or a **WORKSPACE** relative path.

The SET statements in the **[Defines]** section can be used to pre-define macro statements to values that can be used in Conditional Directive statements. The build tools will use these values unless overridden by values on the build tool's command line. When tools process these statements, if no value is given, the tools will assume that the MACRO is either undefined (for the !ifdef and !ifndef conditional directives) or False (for the !if and !elseif conditional directives).

The !include statement may be used in a **[Defines]** section.

Prototype

```
<defines>              ::=  "[defines" [<Attr>] "]" <EOL>
                            [<Statements>]+

<Attr>                 ::=  "." <Arch> ["," "defines." <Arch>]*

<Arch>                 ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                   ::=  [A-Z][A-Z0-9]*

<Statements>           ::=  "DSC_SPECIFCIATION" "=" <SpecValue> <EOL>
                            "PLATFORM_NAME" "=" <Word> <EOL>
                            "PLATFORM_GUID" "=" <RegistryFormatGUID> <EOL>
                            "PLATFORM_VERSION" "=" <String> <EOL>
                            ["SKUID_IDENTIFIER" "=" <SkuUiName> <EOL>]
                            "SUPPORTED_ARCHITECTURES" "=" <ArchList> <EOL>
                            "BUILD_TARGETS" "=" <BuildTargetList> <EOL>
                            "OUTPUT_DIRECTORY" "=" <PATH> <EOL>
                            ["FLASH_DEFINITION" "=" <Filename> <EOL>]
                            ["BUILD_NUMBER" "=" <Digit>{1,4} <EOL>]
                            ["RFC_LANGUAGES" "=" <Rfc4646LangList> <EOL>]
                            ["ISO_LANGUAGES" "=" <Iso6392LangList> <EOL>]
                            [<AddressStatements>]
                            ["DEFINE" <MacroName> "=" {<PATH>} {<VALUE>} <EOL>]*
                            [<SetStatements>] <EOL>]
                            [<IncludeStatements>]

<SpecValue>            ::=  "0x00010015"

<SkuUiName>            ::=  <Word> ["|" <Word>]*

<ArchList>             ::=  <Arch> ["|" <Arch>]*

<AddressStatements>    ::= "FIX_LOAD_TOP_MEMORY_ADDRESS" "=" <Address>

<Address>              ::=  "0x" [<HexDigit>]{8,16}

<NumValue>             ::=  {<Integer>} {<HexNumber>}

<BuildTargetList>      ::=  <Target> ["|" <Target>]*

<Filename>             ::=  [<PATH>] <Word> ["." <Extension>]

<Extension>            ::=  [a-zA-Z][a-zA-Z0-9]*

<PATH>                 ::=  [{"$(WORKSPACE)/"} {<AbsPath>}] [<Path>]*

<AbsPath>              ::=  {"$(WORKSPACE)/"} {"$(EDK_SOURCE)/"} {<SysEnv>}
                            {"$(EFI_SOURCE)/"} {[a-zA-Z]":\"} {"\"} {"/"}

<Path>                 ::=   <Word> "/"
```

```
<File>                ::=  [<PATH>] <Word> ["." <Extension>]

<Rfc4646LangList>     ::=  '"'<Rfc4646Code> [";" <Rfc4646Code>]*'"'

<Iso6392LangList>     ::=  '"'<Iso639-2Code> [<Iso639-2Code>]*'"'

<Rfc4646Code>         ::=  RFC4646 Format Language code

<Iso639-2Code>        ::=  ISO 639-2 Format Language code

<MACRO>               ::=  <Word>

<SetStatements>       ::=  ["SET" <PcdName> "=" [<VALUE>] <EOL>]+

<VALUE>               ::=  {<Boolean>} {<Number>} {<QuotedString>} {<MACRO>}
                           {<Expression>}

<PcdName>             ::=  <TokenSpaceGuidCName> "." <PcdCName>

<TokenSpaceGuidCName> ::=  <Cname>

<PcdCName>            ::=  <Cname>

<Boolean>             ::=  {"TRUE"} {"FALSE"} {"True"} {"False"} {"true"}
                           {"false"}

<Number>              ::=  {<Decimal>} {<Hex>}

<Decimal>             ::=  [{"-"} {"+"}] [0-9]{1,} [["."] [0-9]{0,}]

<Hex>                 ::=  "0x" [a-fA-F0-9]{1,}

<PcdName>             ::=  <TokenSpaceGuidCName> "." <TokenCName>

<TokenCName>          ::=  <CName>

<TokenSpaceGuidCName> ::=  <CName>

<QuotedString>        ::=  {<SingleQuotes>} {<DoubleQuotes>}

<SingleQuotes>        ::=  "'" <String> "'"

<DoubleQuotes>        ::=  '"' <String> '"'

<IncludeStatements>   ::=  "!include" <File>
```

Parameters

SkuUiName

If specified, the image created from the DSC/FDF file pair will only be valid for the SkuUiNames listed. If not specified, and a **[SKUIDS]** section is defined, the image is valid for all SkuIds listed in a **[SKUIDS]** section.

Set Statement PcdName

When a set statement assigns a Macro name to a PCD name, the PCD name does NOT need to be set previously. The PCD name must appear in only one of the PCD

sections in order for the Macro to take on the value of the PCD. The PCD must be defined prior to using the Macro in a directive statement or other statement that permits the use of a Macro name.

RFC4646 Language Code

One or more language codes, formatted per RFC4646, using a semi-colon list separator (example: "en;en-US;es;es-419;fr;fr-FR;zh-Hans-CN".) This list can be used to filter the output of tools that generate unicode strings. For example, if the unicode strings file defines 50 different languages for each string, but the platform integrator only wants to support only three languages, then specifying the three language codes in this statement will limit the final output of string parsing tools to strings for these three languages. Tools must use a "Get Best Language" function when filtering the content. The RFC_LANGUAGES statement should be used when processing EDK II Modules. Space characters are not permitted within the list.

ISO 639-2 Language Code

One or more three character language codes, formatted per ISO 639-2, with no separator character (for example: "engfraspa".) This list can be used to filter the output of tools that generate unicode strings. Tools must use a "Get Best Language" function when filtering the content. The ISO_LANGUAGES statement should be used when processing EDK Components. Space characters are not permitted within the list.

Example

```
[Defines]
  PLATFORM_NAME                  = NT32
  PLATFORM_GUID                  = EB216561-961F-47EE-9EF9-CA426EF547C2
  PLATFORM_VERSION               = 0.3
  DSC_SPECIFICATION              = 0x00010015
  OUTPUT_DIRECTORY               = $(WORKSPACE)/Build/Nt32
  SUPPORTED_ARCHITECTURES        = IA32
  BUILD_TARGETS                  = DEBUG|RELEASE
  RFC_LANGUAGES                  = "en-us;zh-hans;fr-fr"
  ISO_LANGUAGES                  = "engchnfra"
  SKUID_IDENTIFIER               = SkuTwo|DEFAULT
```

3.5 [BuildOptions] Section

Summary

Defines the `[BuildOptions]` section tag which is optional in all EDK II DSC Files.

Use of the `!include` statement in the BuildOptions section is permitted, though not recommended.

This section is used to append flags to then end of the tool code flags defined in the tools_def.txt file. The Family tag can be used for elements that are shared between different architectures, and different tool chain tag names. An example using the Family tag follows:

```
MSFT:*_*_*_CC_FLAGS            = /D MDEPKG_NDEBUG
```

- An optional, special Family tag can be used at the start of the command line, using a colon ":" character after the Family tag. If not specified, specific tool chain tags must be specified (the use of the asterisk "*" wild card character is not permitted.)

Note: *The following is an example which does not use the Family tag, and specific to a specific tool chain tag name:* `RELEASE_MYTOOLS_IA32_CC_FLAGS = /nologo /c /WX /GS- /W4 /Gs8192 /Gy`*.*

- Two types of assignment operators are permitted, the single equal "=" sign is used to append the string to the existing definition, while the double equal "==" sign is used to override any previous definition, replacing the content with just the string.

Prototype

```
<BuildOptions>       ::=  "[BuildOptions" [<Arch> [<EdkType>]] "]" <EOL>
                          [<DefineStatements>]*
                          [<IncludeStatements]*
                          [<Statement>]+

<Arch>               ::=  "." {"Common"} {"IA32"} {"IPF"} {"X64"} {"EBC"} {<OA>}

<OA>                 ::=  [A-Z][A-Z0-9]*

<EdkType>            ::=  "." {"Common"} {"EDK"} {"EDKII"}

<IncludeStatement>   ::=  "!include" <Filename> <EOL>

<DefineStatements>   ::=  "DEFINE" <MACRO> "=" <Value> <EOL>

<Filename>           ::=  [<PATH>] <Word> ["." <Extension>]

<PATH>               ::=  ["$(WORKSPACE)/"] [<Path>]*

<Path>               ::=  <Word> "/"

<Statement>          ::=  {<ToolFlag>} {<ToolPath>} {<ToolCmd>} {<Other>}

<ToolFlag>           ::=  [<Family> ":"] <FlagSpec> <Equal> <Flags> <EOL>

<ToolPath>           ::=  [<Family> ":"] <PathSpec> <Equal> <PATH> <EOL>

<ToolCmd>            ::=  [<Family> ":"] <CmdSpec> "==" <PATH> <ExecCmd> <EOL>

<Other>              ::=  [<Family> ":"] <OtherSpec> <Equal> <String> <EOL>

<Equal>              ::=  {"="} {"=="}

<Family>             ::=  {"MSFT"} {"GCC"} {"INTEL"} {"ACPICA"} {<UserDef>}

<UserDef>            ::=  [A-Z][A-Z0-9]*

<ToolSpec>           ::=  <Target> "_" <TagName> "_" <TargetArch> "_" <ToolCode>

<FlagSpec>           ::=  <ToolSpec> "_FLAGS"

<PathSpec>           ::=  <ToolSpec> "_DPATH"

<CmdSpec>            ::=  <ToolSpec> "_PATH"

<OtherSpec>          ::=  <ToolSpec> "_" <Attribute>

<TargetArch>         ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"*"} {<OA>}
```

```
<Target>          ::=  Target

<Attribute>       ::=  AttributeName

<TagName>         ::=  TagName

<ToolCode>        ::=  ToolCode

<Flags>           ::=  FlagString
```

Parameters

Target

Must match a target identifier in the EDK II tools_def.txt file - the first field, where fields are separated by the underscore character. The "*" character is a valid wild-card.

TagName

Must match a tag name field in the EDK II tools_def.txt file – second field. The "*" character is a valid wild-card.

TargetArch

Must match the architecture field in the EDK II tools_def.txt file – third field. The "*" character is a valid wild-card.

ToolCode

Must match a tool code field in the EDK II tools_def.txt file – fourth field. Use of a wild-card character is not permitted.

AttributeName

Must match a tool attribute field in the EDK II tools_def.txt file – fifth field. Use of a wild-card character is not permitted. The attributes, _Flag, _PATH and DPATH are defined elsewhere and cannot be used with the <OtherSpec> definition.

FlagString

Must be a valid string for the tool specified. The string will be appended to the end of the tool's flags (from the tools_def.txt). Both Microsoft and GCC evaluate options from left to right on the command line. This allows disabling some flags that may have been specified in the tools_def.txt by providing an alternate flag, i.e., if the tools_def.txt file's CC FLAGS defines /O2 and an /O1 options is specified for this module, the module will compile with /O1 (size) not with /O2 (speed). Use of the quote characters around options is required when specifying string values with spaces, path names with spaces or values containing the hash "#" character not within a string. Note that a macro named MDEPKG_NDEBUG is reserved for size reduction purposes. The user should not use this keyword to define new macro.

Example

```
[BuildOptions]
  *_WINDDK3790x1830_*_CC_FLAGS    = /Qwd1418,810
  *_MYTOOLS_*_CC_FLAGS            = /Qwd1418,810
  *_VS2003_*_CC_FLAGS             = /wd4244
  *_WINDDK3790x1830_*_CC_FLAGS    = /wd4244
  *_MYTOOLS_*_CC_FLAGS            = /wd4244
# Flags used by Dynamic linker
  DEBUG_WINDDK3790x1830_IA32_DLINK_FLAGS = \
       /EXPORT:InitializeDriver=_ModuleEntryPoint /ALIGN:4096 \
       /SUBSYSTEM:CONSOLE
```

3.6 [SkuIds] Section

Summary

Defines the **[SkuIds]** tag which is optional in all EDK II DSC files. Entries may appear in any order. This section lists numeric mappings to the SKU ID User Interface Name, only valid values from this list can be specified in the defines section. Only one **!include** statement is supported; it should use a ".txt" extension, and the include filename may include the variable, **$(WORKSPACE)**, otherwise it must be located in the current working directory.

If this section is not specified, the parsing tools will assume a SkuID of 0, with a UiName of "**DEFAULT**."

It is recommended that the file should be stored in the **$(WORKSPACE)/Conf** directory.

Only one **!include** statement may be used within the SkuID section.

The **!include** file can only contain an ASCII (not Unicode) list of "**integer|UiSkuName**" statements.

This is an optional section for EDK II DSC files.

Prototype

```
<SkuId>              ::=  "[SkuIds]" <EOL>
                          {<Statement>} {<IncludeStatement>}

<Statement>          ::=  [<Integer> "|" <UiName> <EOL>]+

<IncludeStatement>   ::=  "!include" <Filename> <EOL>

<Filename>           ::=  [<PATH>] <Word> [".txt"]

<PATH>               ::=  ["$(WORKSPACE)/"] <Path>

<Path>               ::=  [<Word> "/"]*

<UiName>             ::=  <Word>
```

Example

```
[SkuIds]
  0|DEFAULT   # The entry: 0|DEFAULT is reserved and always required.
  1|SkuTwo
```

3.7 [Libraries] Section

Summary

Defines the `[Libraries]` section tag which lists the libraries that are linked against EDK I components. Entries may appear in any order. Entries for EDK are a list of INF files, with a path that is *always* relative to the `$(EDK_SOURCE)` directory (or a MACRO definition).

One or more `!include` statements may be used within the libraries sections. If used, the file included must be a list of INF files and their paths relative to the `$(EDK_SOURCE)` directory.

This section is an optional section in EDK II DSC files, although if any EDK component is specified in the `[Components]` section, then the EDK II DSC file must contain this section. EDK components can not use EDK II Library Instances.

Prototype

```
<Libraries>          ::=  "[Libraries" [<attrs>] "]" <EOL>
                          [<DefineStatements>]*
                          [<IncludeStatement>]*
                          [<Statement>]*

<attrs>              ::=  "." <arch> [", libraries." <arch>]*

<arch>               ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"Common"}

<IncludeStatement>   ::=  "!include" <Filename> <EOL>

<Filename>           ::=  [<PATH>] <Word> [".txt"]

<Statement>          ::=  <PathAndFilename> <EOL>

<PathAndFilename>    ::=  <PATH> <Word> ".inf"

<PATH>               ::=  ["$(EDK_SOURCE)" "/"] [<Path>]*

<Path>               ::=  <Word> "/"

<DefineStatements>   ::=  "DEFINE" <MACRO> "=" <PATH> <EOL>

<MACRO>              ::=  <Word>
```

Example

```
[Libraries]
Foundation/Efi/Guid/EfiGuidLib.inf
Foundation/Framework/Guid/EdkFrameworkGuidLib.inf
Foundation/Guid/EdkGuidLib.inf
Foundation/Library/EfiCommonLib/EfiCommonLib.inf
Foundation/Cpu/Pentium/CpuIA32Lib/CpuIA32Lib.inf
Foundation/Cpu/Itanium/CpuIA64Lib/CpuIA64Lib.inf
Foundation/Library/CustomizedDecompress/CustomizedDecompress.inf

[Libraries.IA32]
DEFINE PLATFORM_DIR = $(EDK_SOURCE)/Platform
$(PLATFORM_DIR)/IntelEpg/Guid/IntelEpgGuidLib.inf
$(PLATFORM_DIR)/IntelEpg/Ppi/IntelEpgPpiLib.inf
$(PLATFORM_DIR)/Generic/Guid/GenericGuidLib.inf
$(PLATFORM_DIR)/Generic/Lib/Port80MappingLib/PlatformPort80.inf
```

3.8 [LibraryClasses] Section

Summary

This section defines the `[LibraryClasses]` tag required for EDK II module INF files, and is new for EDK II extended DSC files. This is a mapping of library class names to the EDK II module instances that provide the library class.

This is an optional EDK II section, if no library classes are defined for any of the components, or if only EDK modules are used.

The one or more `!include` statements may be used within the library class section.

The `!include` files may contain `LibraryClass | Library Instance` statements as well as complete Library Class sections.

The library class entry is a formatted string with two fields, separated by the pipe "|" character.

When parsing the DSC file, the precedence rules should be followed.

1. If a Library Class Instance (INF) is specified in the EDK II `[Components]` section (INF file's `<LibraryClasses>` sub-section,) then it will be used.
2. If not specified in the make `[Components]` Section, then the Library Class Instance that is defined in the `[LibraryClasses.$(ARCH).$(MODULE_TYPE)]` Section should be used.
3. If not specified in the `[LibraryClasses.$(ARCH).$(MODULE_TYPE)]` Section, then the Library Class Instance that is defined in the `[LibraryClasses.Common.$(MODULE_TYPE)]` Section should be used.
4. If not specified in the `[LibraryClasses.Common.$(MODULE_TYPE)]` Section, then the Library Class Instance that is defined in the `[LibraryClasses.$(ARCH)]` Section should be used.
5. If not specified in the `[LibraryClasses.$(ARCH)]` Section, then the Library Class Instance that is defined in the `[LibraryClasses]` Section or `[LibraryClasses.Common]` Section should be used.
6. It is an error if it has not been specified in one of the above sections.

Prototype

```
<libraryclasses>        ::=  "[LibraryClasses" [<Attrs>] "]" <EOL>
                             [<DefineStatements>]*
                             [<IncludeStatement>]*
                             <LcEntry>+

<Attrs>                 ::=  <SubAttr> ["," "LibraryClasses" <SubAttr>]*

<SubAttr>               ::=  "." <arch>  ["." <ModuleType>]

<ModuleType>            ::=  {"BASE"} {"SEC"} {"PEI_CORE"} {"PEIM"} {"DXE_CORE"}
                             {"DXE_DRIVER"} {"DXE_SMM_DRIVER"} {"SMM_CORE"}
                             {"DXE_SAL_DRIVER"} {"DXE_RUNTIME_DRIVER"}
                             {"UEFI_DRIVER"} {"UEFI_APPLICATION"}

<arch>                  ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                    ::=  [A-Z][A-Z0-9]*

<DefineStatements>      ::=  "DEFINE" <WORD> "=" <PATH> <EOL>

<IncludeStatement>      ::=  "!include" <Filename> <EOL>

<Filename>              ::=  [<PATH>] <Word> [".txt"]

<LcEntry>               ::=  <LibraryClassKeyWord> "|" <LibraryInstance> <EOL>

<LibraryClassKeyWord>   ::=  <Word>

<LibraryInstance>       ::=  <PATH> <InfFileName> <EOL>

<InfFileName>           ::=  <Word> ".inf"

<PATH>                  ::=  ["$(WORKSPACE)" "/"] [<Word> "/"]*
```

Example

```
[libraryclasses.common]
BaseLib|MdePkg/Library/BaseLib.inf
BaseMemoryLib|MdePkg/Library/BaseMemoryLib/BaseMemoryLib.inf
DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
PcdLib|MdePkg/Library/BasePcdLib/BasePcdLibNull.inf

[libraryclasses.common.PEI_CORE, libraryclasses.common.PEIM]
DEFINE MDIR = MdePkg/Library
DebugLib|$(MDIR)/PeiReportStatusCode/PeiReportStatusCodeLib.inf

[libraryclasses.common.UEFI_DRIVER, libraryclasses.common.UEFI_APPLICATION]
DebugLib|MdePkg/Library/UefiDebugLibStdErr/UefiDebugLibStdErr.inf
```

3.9 PCD Sections

PCD Values listed in the DSC file may be absolute values, macro names or expressions which may include other PCD names and/or macro names.

3.9.1 [Pedestrianisms] Section

This is an optional section for EDK II DSC Files.

Summary

Defines the `[PcdsFeatureFlag]` section tag that may be required for platform DSC files that will use EDK II module INF files.

Each entry must be on a single line.

The `!include` statement is permitted in the `[PcdsFeatureFlag]` sections, although not recommended.

The PcdsFeatureFlag entry is a formatted string with two fields, separated by the pipe character, "|".

Prototype

```
<PcdFeatureFlags>       ::=  "[PcdsFeatureFlag" [<Attrs>] "]" <EOL>
                             [<IncludeStatement>]
                             <PcdEntry>+

<Attrs>                 ::=  <ArchArg> [", PcdsFeatureFlag" <ArchArg>]*

<ArchArg>               ::=  "." <Arch>

<Arch>                  ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                    ::=  [A-Z][A-Z0-9]*

<IncludeStatement>      ::=  "!include" <Filename> <EOL>

<Filename>              ::=  [<PATH>] <Word> [".txt"]

<PATH>                  ::=  ["$(WORKSPACE)" "/"] [<Path>]*

<Path>                  ::=  <Word> "/"

<PcdEntry>              ::=  <PcdName> "|" <PcdValue> [<comment>] <EOL>

<PcdName>               ::=  <TokenSpaceGuidCName> "." <TokenCName>

<TokenCName>            ::=  <CName>

<TokenSpaceGuidCName>   ::=  <CName>

<PcdValue>              ::=  {<BoolType>} {<MACRO>} {<BoolExpression>}

<MACRO>                 ::=  "$(" <WORD> ")"

<WORD>                  ::=  (A-Z)[(A-Z0-9_)]*

<BoolExpression>        ::=  Any combination of C style expressions using
                             numeric values, macros or other PCD names that
                             evaluate to either TRUE or FALSE
```

Parameters

BoolExpression

C-style expression using C relational, equality and logical numeric and bitwise operators that evaluate to either TRUE (1) or FALSE (0). Values other than zero or one are invalid. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

Example

```
[PcdsFeatureFlag]
  gEfiMdePkgTokenSpaceGuid.PcdComponentNameDisable|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxePcdDatabaseTraverseEnabled|TRUE
```

3.9.2 [PcdsFixedAtBuild] Section

Summary

This is an optional section for EDK II DSC Files.

Each entry must be on a single line.

This section defines the `[PcdsFixedAtBuild]` section tag that may be required for EDK II DSC files that will use EDK II module INF files. The values listed below must match the datum type specified in the DEC file that declares this PCD. Note that if a PCD's datum type is `VOID*` the `MaximumDatumSize` field is required.

The !include statement is permitted in `[PcdsFixedAtBuild]` sections, although not recommended.

The `[PcdsFixedAtBuild]` entry is a formatted string consisting of two to four fields that are separated by the pipe character, "|".

Prototype

```
<pcd>                    ::=  [PcdsFixedAtBuild [<Args>] ] <EOL>
                              [<DefineStatements>]*
                              [<IncludeStatement>]
                              <PcdEntry>+

<Args>                   ::=  <Arch> [", PcdsFixedAtBuild" <Arch>]*

<ArchArg>                ::=  "." <Arch>

<Arch>                   ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                     ::=  [A-Z][A-Z0-9]*

<IncludeStatement>       ::=  "!include" <Filename> <EOL>

<DefineStatemetns>       ::=  "DEFINE" <MACRO> "=" <Value> <EOL>

<Filename>               ::=  [<PATH>] <Word> [".txt"]

<PATH>                   ::=  ["$(WORKSPACE)/"] [<Path>]*

<Path>                   ::=  <Word> "/"

<PcdEntry>               ::=  <PcdName> "|" <PcdValue> [<comment>] <EOL>

<PcdName>                ::=  <TokenSpaceGuidCName> "." <TokenCName>

<TokenName>              ::=  <CName>

<TokenSpaceGuidCName>    ::=  <CName>

<PcdValue>               ::=  {<Boolean>} {<Number>} {<StringData>} {<MACRO>}
                              {<Expression>}

<MACRO>                  ::=  "$(" <WORD> ")"

<Boolean>                ::=  {<BoolType>} {<BoolExpression>} ["| BOOLEAN"]

<Number>                 ::=  {"0x" <HexDigit>*} {<Integer>} {<Expression>}
                              ["|" <NumType>]

<NumType>                ::=  {UINT8} {UINT16} {UINT32} {UINT64}

<StringData>             ::=  <StrValue> ["| VOID* |" <MaxDatumSize>] <EOL>

<StrValue>               ::=  {<StringType>} {<DataArray>}

<StringType>             ::=  {<AsciiString>} {<UnicodeString>} {<TOKEN>}

<DataArray>              ::=  "{" <HexValue> ["," <HexValue>]* "}"

<MaxDatumSize>           ::=  {"0x" <HexDigit>*} {<Integer>} {<Expression>}
```

Parameters

BoolExpression

C-style expression using C relational, equality and logical numeric and bitwise operators that evaluate to either TRUE (1) or FALSE (0). Values other than zero or one are invalid. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

Expression

C-style expression using C relational, equality and logical numeric and bitwise operators and/or arithmetic and bitwise operators that evaluate to a value that matches the Datum Type of the PCD. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

UEFI String Token Name | Language Code

This is a non-unicode version of an HII string token name, obtained from UNI files. Tools must provide a method of looking up the value of string, using the token name (converted to unicode by the tools) from the UCS2 file. Either the RFC4646 or the ISO-639-2 language code must be provided. If the code is not defined in the UCS2 file, then the tools must provide a best match, with a default of "en" if no best match is found.

Example

```
[PcdsFixedAtBuild.IA32]
  gEfiMdePkgTokenSpaceGuid.PcdFSBClock|200000000|UINT64
  gEfiMdePkgTokenSpaceGuid.PcdMaximumUnicodeStringLength|1000000
  gEfiEdkNt32PkgTokenSpaceGuid.PcdWinNtMemorySizeForSecMain| L"64!64" | \
    10 | VOID* | 16
```

3.9.3 [PcdsPatchableInModule] Section

Summary

Defines the **[PcdsPatchableInModule]** section tag that may be required for EDK II DSC files that will use EDK II module INF files. The values listed below must match the datum type specified in the DEC file that declares this PCD. Note that if a PCD's datum type is **VOID*** the **MaximumDatumSize** field is required.

This is an optional section for EDK II DSC Files.

An entry must be on a single line.

The **!include** statement is permitted in **[PcdsPatchableInModule]** sections, although not recommended.

The **[PcdsFixedAtBuild]** and/or **[PcdsPatchableInModule]** entry is a formatted string with two to four fields, separated by the pipe | character.

Prototype

```
<pcd>                  ::=  "[PcdsPatchableInModule" [<Args>] "]" <EOL>
                            [<DefineStatements>]*
                            [<IncludeStatement>]
                            <PcdEntry>+

<Args>                 ::=  <Arch> ["," "PcdsPatchableInModule" <Arch>]*

<Arch>                 ::=  "." <Arches>

<Arches>               ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                   ::=  [A-Z][A-Z0-9]*

<IncludeStatement>     ::=  "!include" <Filename> <EOL>

<DefineStatements>     ::=  "DEFINE" <MACRO> "=" <Value> <EOL>

<Filename>             ::=  [<PATH>] <Word> [".txt"]

<PATH>                 ::=  ["$(WORKSPACE)/"] [<Path>]*

<Path>                 ::=  <Word> "/"

<PcdEntry>             ::=  <PcdName> "|" <PcdValue> [<comment>] <EOL>

<PcdName>              ::=  <TokenSpaceGuidCName> "." <TokenCName>

<TokenName>            ::=  <CName>

<TokenSpaceGuidCName>  ::=  <CName>

<PcdValue>             ::=  {<Boolean>} {<Number>} {<StringData>} {<MACRO>}
                            {<Expression>}

<MACRO>                 ::=  "$(" <WORD> ")"

<Boolean>              ::=  {<BoolType>} {<BoolExpression>} ["|" "BOOLEAN"]

<Number>               ::=  {"0x" <HexDigit>*} {<Integer>} {<Expression>}
                            ["|" <NumType>]

<NumType>              ::=  {"UINT8"} {"UINT16"} {"UINT32"} {"UINT64"}

<StringData>           ::=  <StrValue> ["| VOID* |" <MaxDatumSize>] <EOL>

<StrValue>             ::=  {<StringType>} {<DataArray>}

<StringType>           ::=  {<AsciiString>} {<UnicodeString>} {<TOKEN>}

<DataArray>            ::=  "{" <HexValue> ["," <HexValue>]* "}"

<MaxDatumSize>         ::=  {"0x" <HexDigit>*} {<Integer>} {<Expression>}

<TOKEN>                ::=  <UEFI_String_Token_Name> "|" <LanguageCode>
```

Parameters

BoolExpression

C-style expression using C relational, equality and logical numeric and bitwise operators that evaluate to either TRUE (1) or FALSE (0). Values other than zero or one are invalid. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

Expression

C-style expression using C relational, equality and logical numeric and bitwise operators and/or arithmetic and bitwise operators that evaluate to a value that matches the Datum Type of the PCD. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

UEFI String Token Name | Language Code

This is a non-unicode version of an HII string token name, obtained from UNI files. Tools must provide a method of looking up the value of string, using the token name (converted to unicode by the tools) from the UCS2 file. Either the RFC4646 or the ISO-639-2 language code must be provided. If the code is not defined in the UCS2 file, then the tools must provide a best match, with a default of "en" if no best match is found.

Example

```
[PcdsPatchableInModule]
  gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000000
```

3.9.4 [PcdsDynamic] Section

This is an optional section for EDK II DSC Files.

Summary

Defines the **[PcdsDynamic]** section tag that may be required for platform DSC files that will use EDK II module INF files. The values listed below must match the datum type specified in the DEC file that declares this PCD. Note that if a PCD's datum type is **VOID*** the **MaximumDatumSize** field is required. Note that specifying different **<DataStore>** values (**Default**, **HII**, or **VPD**) in one section header is not permitted. It is permissible to specify different Architectures using an identical **<DataStore>** value. The following line is permissible:

```
[PcdsDynamicDefault.X64.Default, PcdsDynamic.IPF.Default]
```

The following line is incorrect, and should cause a build break during parsing of the file:

```
[PcdsDynamic.X64.Default, PcdsDynamicVpd.IPF.Default]
```

An entry must be on a single line.

The **!include** statement is permitted in **[PcdsDynamic]** sections, although not recommended.

The **[PcdsDynamic]** entry is a formatted string consisting of three to six fields that are separated by the pipe character, "|".

Prototype

```
<pcd>                  ::=  "[" <PcdTypes> [<Args>] "]" <EOL>
                            [<DefineStatements>]*
                            [<IncludeStatement>]
                            <PcdEntry>+

<PcdTypes>             ::=  {"PcdsDynamic"} {"PcdsDynamicDefault"}
                            {"PcdsDynamicHii"} {"PcdsDynamicVpd"}

<Args>                 ::=  <Primative> ["," <PcdTypes> <Primative>]*

<Primative>            ::=  "." <Arch> [<SkuId>]

<Arch>                 ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                   ::=  [A-Z][A-Z0-9]*

<DataStore>            ::=  {<default>} {<hii>} {<vpd>}

<SkuId>                ::=  "." <SkuIdentifier>

<default>              ::=  "Default"

<hii>                  ::=  "Hii"

<vpd>                  ::=  "Vpd"

<SkuIdentifier>        ::=  <Keyword> ["|" <Keyword>]*

<DefineStatements>     ::=  "DEFINE" <MACRO> "=" <Value> <EOL>

<IncludeStatement>     ::=  "!include" <Filename> <EOL>

<Filename>             ::=  [<PATH>] <Word> [".txt"]

<PATH>                 ::=  ["$(WORKSPACE)/"] [<Path>]*

<Path>                 ::=  <Word> "/"

<PcdEntry>             ::=  IF <DataStore> == <hii>
                              <hiiEntry>+
                            IF <DataStore> == <vpd>
                              <vpdEntry>+
                            ELSE
                              <minEntry1>+

<minEntry1>            ::=  <PcdName> "|" <Value> <EOL>

<PcdName>              ::=  <TokenSpaceGuidCName> "." <PcdCName>

<PcdCName>             ::=  <CName>

<TokenSpaceGuidCName>  ::=  <CName>

<Value>                ::=  {<Boolean>} {<Number>} {<String>} {<DataArray>}
```

```
                                {<MACRO>}

<MACRO>                 ::=  "$(" <WORD> ")"

<Boolean>               ::=  {<BoolType>} {<BoolExpression>} ["|" BOOLEAN]

<Number>                ::=  { "0x" [a-fA-F0-9]+ } { [0-9]+ } {<Expression>}

<String>                ::=  {<StringType>} [<DatumTypeSize>]

<StringType>            ::=  {<AsciiString>} {<UnicodeString>} {<TOKEN>}

<DatumTypeSize>         ::=  "| VOID*" [<MaxDatumSize>]

<MaxDatumSize>          ::=  "|" {"0x" <HexDigit>*} {<Integer>} {<Expression>}

<DataArray>             ::=  <HexArray> [<DatumTypeSize>]

<HexArray>              ::=  "{" "0x" <HexDigit>+ ["," "0x" <HexDigit>+]* "}"

<vpdEntry>              ::=  <PcdName> "|" <VpdOffset> [<VpdOptionalData>] <EOL>

<vpdOptionalData>       ::=  [<MaxDatumSize> [<VpdInitialValue>]]

<VpdInitialValue>       ::=  "|" <DefaultValue>

<VpdOffset>             ::=  {"0x" <HexDigit>+} {<Integer>}

<hiiEntry>              ::=  <PcdName> <Field2>

<Field2>                ::=  "|" <HiiString> <Field3>

<HiiString>             ::=  {["0x" [a-fA-F0-9]{4} " "]+} {"L" <QuotedString>}
                             {<TOKEN>}

<Field3>                ::=  "|" <VariableGuidCName> <Field4>

<VariableGuidCName>     ::=  <CName>

<Field4>                ::=  "|" <VariableOffset> ["|" <DefaultValue>] <EOL>

<VariableOffset>        ::=  {"0x" <HexDigit>+} {<Integer>} {<Word> "." <Word>}

<DefaultValue>          ::=  {<Boolean>} {<Number>} {<String>} {<DataArray>}
                             {<MACRO>}

<TOKEN>                 ::=  <UEFI_String_Token_Name> "|" <LanguageCode>
```

Parameters

BoolExpression

C-style expression using C relational, equality and logical numeric and bitwise operators that evaluate to either TRUE (1) or FALSE (0). Values other than zero or one are invalid. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For

both macro names and PCDs, the element must be previously defined before it can be used.

Expression

C-style expression using C relational, equality and logical numeric and bitwise operators and/or arithmetic and bitwise operators that evaluate to a value that matches the Datum Type of the PCD. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

VpdOffset

A non-negative numeric value that is the number of bytes from the start of the Vpd Region specified in the FDF file.

VariableOffset

A non-negative numeric value that is the number of bytes from the start to the start of this variable.

UEFI String Token Name | Language Code

This is a non-unicode version of an HII string token name, obtained from UNI files. Tools must provide a method of looking up the value of string, using the token name (converted to unicode by the tools) from the UCS2 file. Either the RFC4646 or the ISO-639-2 language code must be provided. If the code is not defined in the UCS2 file, then the tools must provide a best match, with a default of "en" if no best match is found.

Examples

```
[PcdsDynamicDefault.IA32.Default]
  gEfiEdkNt32PkgTokenSpaceGuid.PcdWinNtMemorySizeForSecMain|"L64!64"|10
  gEfiCpuTokenSpaceGuid.PcdCpuProcessorFeatureUserConfiguration | \
    0xffffffff
  gEfiCpuTokenSpaceGuid.PcdCpuProcessorFeatureCapability|0

[pcdsDynamicVpd.common.Sku_Two|SkuSeven]
  gEfiCpuTokenSpaceGuid.PcdProcessorFeatureUserConfiguration | \
    0x0001ffff | 32

[pcdsDynamicHii.IA32, pcdsDynamicHii.X64.Sku1]
  gEfiMyModulePkgTokenSpaceGuid.PcdChassisIntrution | \
    0x0053 0x0065 0x0074 0x0075 0x0070|gSysConfigGuid|0x83|0x0
```

3.9.5 [PcdsDynamicEx] Section

Summary

Defines the **[PcdsDynamicEx]** section tag that may be required for platform DSC files that will use EDK II module INF files. The values listed below must match the datum type specified in the DEC file that declares this PCD. Note that if a PCD's datum type is **VOID*** the **MaximumDatumSize** field is required. Note that specifying different **<DataStore>** values (**Default**, **HII**, or **VPD**) in one section header is not permitted. It is permissible to specify different Architectures using an identical **<DataStore>** value. The following line is permissible:

```
[PcdsDynamicEx.X64.Default, PcdsDynamicExDefault.IPF.Default]
```

The following line is incorrect, and should cause a build break during parsing of the file:

`[PcdsDynamicExDefault.X64.DEFAULT, PcdsDynamicExVpd.IPF.DEFAULT]`

This is an optional section for EDK II DSC Files, and not valid for EDK DSC files.

An entry must be on a single line. Multiple entries on one line and entries spanning multiple lines are prohibited.

The `!include` statement is permitted in `[PcdsDynamicEx]` sections, although not recommended.

The `[PcdsDynamicEx]` entry is a formatted string with three to six fields, separated by the pipe `|` character.

Prototype

```
<pcd>                  ::=  "[" <PcdTypes> [<Args>] "]" <EOL>
                            [<DefineStatements>]*
                            [<IncludeStatement>]
                            <PcdEntry>+

<PcdTypes>             ::=  {"PcdsDynamicEx"} {"PcdsDynamicExDefault"}
                            {"PcdsDynamicExHii"} {"PcdsDynamicExVpd"}

<Args>                 ::=  <Primative> ["," <PcdTypes> <Primative>]*

<Primative>            ::=  "." <Arch> [<SkuId>]

<Arch>                 ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"common"} {<OA>}

<OA>                   ::=  [A-Z][A-Z0-9]*

<DataStore>            ::=  {<default>} {<hii>} {<vpd>}

<SkuId>                ::=  "." <SkuIdentifier>

<default>              ::=  "Default"

<hii>                  ::=  "Hii"

<vpd>                  ::=  "Vpd"

<SkuIdentifier>        ::=  <Keyword> ["|" <Keyword>]*

<DefineStatements>     ::=  "DEFINE" <MACRO> "=" <Value> <EOL>

<IncludeStatement>     ::=  "!include" <Filename> <EOL>

<Filename>             ::=  [<PATH>] <Word> [".txt"]

<PATH>                 ::=  ["$(WORKSPACE)/"] [<Path>]*

<Path>                 ::=  <Word> "/"

<PcdEntry>             ::=  IF <DataStore> == <hii>
                              <hiiEntry>+
                            IF <DataStore> == <vpd>
                              <vpdEntry>+
                            ELSE
                              <minEntry1>+

<minEntry1>            ::=  <PcdName> "|" <Value> <EOL>

<PcdName>              ::=  <TokenSpaceGuidCName> "." <PcdCName>

<PcdCName>             ::=  <CName>

<TokenSpaceGuidCName>  ::=  <CName>

<Value>                ::=  {<Boolean>} {<Number>} {<String>} {<DataArray>}
```

```
                                {<MACRO>}

<MACRO>                  ::=  "$(" <WORD> ")"

<Boolean>                ::=  {<BoolType>} {<BoolExpression>} ["|" "BOOLEAN"]

<Number>                 ::=  { "0x" [a-fA-F0-9]+ } { [0-9]+ } {<Expression>}

<String>                 ::=  <StringType> [<DatumTypeSize>]

<StringType>             ::=  {<AsciiString>} {<UnicodeString>} {<TOKEN>}

<DatumTypeSize>          ::=  "|" "VOID*" [<MaxDatumSize>]

<MaxDatumSize>           ::=  "|" {"0x" <HexDigit>*} {<Integer>} {<Expression>}

<DataArray>              ::=  <HexArray> [<DatumTypeSize>]

<HexArray>               ::= "{" "0x" <HexDigit>+ ["," "0x" <HexDigit>+]* }

<vpdEntry>               ::=  <PcdName> "|" <VpdOffset> [<VpdOptionalData>] <EOL>

<VpdOptionalData>        ::=  [<MaxDatumSize> [<VpdInitialValue>]]

<VpdInitialValue>        ::=  "|" <DefaultValue>

<VpdOffset>              ::=  {"0x" <HexDigit>+} {<Integer>}

<hiiEntry>               ::=  <PcdName> <Field2>

<Field2>                 ::=  "|" <HiiString> <Field3>

<HiiString>              ::=  {["0x" [a-fA-F0-9]{4} " "]+} {"L" <QuotedString>}
                              {<TOKEN>}

<Field3>                 ::=  "|" <VariableGuidCName> <Field4>

<VariableGuidCName>      ::=  <CName>

<Field4>                 ::=  "|" <VariableOffset> ["|" <DefaultValue>] <EOL>

<VariableOffset>         ::=  {"0x" <HexDigit>+} {<Integer>} {<Word> "." <Word>}

<DefaultValue>           ::=  {<Boolean>} {<Number>} {<String>} {<DataArray>}
                              {<MACRO>}

<TOKEN>                  ::=  <UEFI_String_Token_Name> "|" <LanguageCode>
```

Parameters

BoolExpression

C-style expression using C relational, equality and logical numeric and bitwise operators that evaluate to either TRUE (1) or FALSE (0). Values other than zero or one are invalid. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For

both macro names and PCDs, the element must be previously defined before it can be used.

Expression

C-style expression using C relational, equality and logical numeric and bitwise operators and/or arithmetic and bitwise operators that evaluate to a value that matches the Datum Type of the PCD. Precedence and associativity follow C standards. Along with absolute values, macro names and PCDs may be used within an expression. For both macro names and PCDs, the element must be previously defined before it can be used.

VpdOffset

A non-negative numeric value that is the number of bytes from the start of the Vpd Region specified in the FDF file.

VariableOffset

A non-negative numeric value that is the number of bytes from the start to the start of this variable.

UEFI String Token Name | Language Code

This is a non-unicode version of an HII string token name, obtained from UNI files. Tools must provide a method of looking up the value of string, using the token name (converted to unicode by the tools) from the UCS2 file. Either the RFC4646 or the ISO-639-2 language code must be provided. If the code is not defined in the UCS2 file, then the tools must provide a best match, with a default of "en" if no best match is found.

Examples

```
[PcdsDynamicExDefault.IA32.Default]
  gEfiEdkNt32PkgTokenSpaceGuid.PcdWinNtMemorySizeForSecMain|L64!64|10
  gEfiCpuTokenSpaceGuid.PcdCpuProcessorFeatureUserConfiguration | \
    0xffffffff
  gEfiCpuTokenSpaceGuid.PcdCpuProcessorFeatureCapability|0

[pcdsDynamicExVpd.common.Sku_Two|SkuSeven]
  gEfiCpuTokenSpaceGuid.PcdProcessorFeatureUserConfiguration | \
    0x0001ffff | 32

[pcdsDynamicExHii.IA32, pcdsDynamicExHii.X64.Sku1]
  gEfiMyModulePkgTokenSpaceGuid.PcdChassisIntrution | \
    0x0053 0x0065 0x0074 0x0075 0x0070|gSysConfigGuid|0x83|0x0
```

3.10 [Components] Section

Summary

Defines the `[Components]` section tag that is required for platform DSC files. This section defines the modules and components that will be processed by compilation tools and the EDK II tools used to generate PE32/PE32+/Coff image files.

This is a required section for EDK II DSC files.

The `!include` statement is permitted in `[components]` sections. however this method is NOT recommended.

All paths must be specified, directly or indirectly, relative to the global environment

variables, **EDK_SOURCE** for EDK components and **WORKSPACE** for EDK II modules. If the environment variable is not specified, the INF file path is assumed to be relative to the **WORKSPACE**.

The following is an example of specifying a Direct Path:

```
$(WORKSPACE)/MdeModulePkg/Universal/Disk/DiskIo/Dxe
```

The following is an example of specifying an Indirect Path:

```
DEFINE DISK = $(WORKSPACE)/MdeModulePkg/Universial/Disk
$(DISK)/DiskIo/Dxe
```

The permitted **DEFINE** statement must be a variable name assigned to a path.

EDK II DSC files allow specifying only one optional argument on an INF file entry line. The argument, EXEC = filename, is used for User Defined processing of an INF file. The EDK II parsing tools will call the program listed by filename (which must either be in the OS PATH environment, or fully qualified path and filename) with the INF filename (expanded) as the one and only argument to filename.

EDK II modules can have scoped sub-elements, **<LibraryClasses>**, **<Pcd*>** and/or **<BuildOptions>** that allow individual modules to supersede previous definitions. The values specified in the sub-elements are given the highest precedence by the parsing tool.

The format for items listed in the sub-elements is the identical format for content under the section.

Within the context of an EDK II module sub-element, the **<LibraryClasses>** entries must appear before **<Pcd*>** entries; the **<LibraryClasses>** entries terminate with the start of either the **<Pcd*>** or **<BuildOptions>** sub-section header. The **<BuildOptions>** sub-element must be the last sub-entry of an EDK II module's scoped section. Entries for **<LibraryClasses>**, **<PCD*>** and **<BuildOptions>** are used to replace the platform or global definition entries listed elsewhere. LibraryClass and PCD*s are globally defined in the DSC file's **[LibraryClasses]** and **[Pcd*]** sections, while global **BuildOptions** may be specified in either the DSC file's **[BuildOptions]** section or in the **$(WORKSPACE)/Conf/tools_def.txt** file.

It should be noted that components and modules listed here will be processed during the MAKE phase of the build. Binary EDK II only modules do not need to be listed in this section, but can be put into the FDF file. Build tools will use the order of files specified in this section for performing a build (Library Class Instances will be built prior to the module's sources,) however the ordering in this file has not affect on the ordering of modules in a binary image (the FDF file describes that ordering.)

Prototype

```
<components>            ::=  "[Components" [<Args>] "]" <EOL>
                               [<DefineStatements>]*
                               [<IncludeStatement>]*
                               [<SetStatements>]*
                               [<InfFiles>]*

<Args>                  ::=  "." <arch> ["," "components." <arch>]*

<arch>                  ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {"Common"} {<OA>}

<OA>                    ::=  [A-Z][A-Z0-9]*

<DefineStatements>      ::=  "DEFINE" <MACRO> "=" {<PATH>} {<Value>} <EOL>

<MACRO>                 ::=  <Word>

<DefineStatements>      ::=  "Set" <PcdName> "=" <Value> <EOL>

<PATH>                  ::=  [{"$(WORKSPACE)/"} {"$(" <MACRO> ")"}] [<Path>]*

<Path>                  ::=  <Word> "/"

<IncludeStatement>      ::=  "!include" <Filename> <EOL>

<Filename>              ::=  [<PATH>] <Word> ["." <Extension>]

<comment>               ::=  "#" [" " <String>]

<InfFiles>              ::=  <InfFilename> [<Options>] <EOL>

<Options>               ::=  {<Exec>} {<Edk2Struct>}

<InfFilename>           ::=  <PATH> <Word> ".inf"

<Exec>                  ::=  "EXEC" "=" <ExecFilename>

<ExecFilename>          ::=  <PATH> <Word> ["." <ExecExtension>]

<ExecExtension>         ::=  <Word> # An OS recognisable extension that will
                                    # automatically be run.

<Edk2Struct>            ::=  { <EOL>
                               [<LibraryClass>]
                               [<PcdsFeatureFlag>]
                               [<PcdsFixedAtBuild>]
                               [<PcdsPatchableInModule>]
                               [<PcdsDynamic>]
                               [<PcdsDynamicEx>]
                               [<BuildOptions>]
                             } <EOL>

<LibraryClass>          ::=  "<LibraryClasses>" <EOL>
                               [<DefineStatements>]*
                               [<LibraryInstances>]+
```

```
<LibraryInstances>      ::=  {<ClassInstanceMap>} {<NullLibraryInstances>}

<LibInstanceMap>        ::=  <ClassName> "|" <InfFilename> <EOL>

<ClassName>             ::=  A Library Class Keyword defined in DEC files.
                             The Keyword must also be present in the defines
                             section LIBRARY_CLASS entry of the INF file.

<NullLibraryInstances>  ::=  "NULL" "|" <InfFilename> <EOL>

<PcdsFeatureFlag>       ::=  "<PcdsFeatureFlag>" <EOL>
                               [<DefineStatements>]*
                               [<SetStatements>]*
                               [<PcdsFeatureFlagEntry>]+

<PcdsFixedAtBuild>      ::=  "<PcdsFixedAtBuild>" <EOL>
                               [<DefineStatements>]*
                               [<SetStatements>]*
                               [<PcdFixedAtBuildEntry>]+

<PcdsPatchableInModule> ::=  "<PcdsPatchableInModule>" <EOL>
                               [<DefineStatements>]*
                               [<SetStatements>]*
                               [<PcdPatchableInModuleEntry>]+

<PcdsDynamic>           ::=  {"<PcdsDynamic>"} {"<PcdsDynamicDefault>"}
                             {"<PcdsDynamicHii>"} {"<PcdsDynamicVpd>"} <EOL>
                               [<DefineStatements>]*
                               [<SetStatements>]*
                               [<PcdsDynamicEntry>]+

<PcdsDynamicEx>         ::=  {"<PcdsDynamicEx>"} {"<PcdsDynamicExDefault>"}
                             {"<PcdsDynamicExHii>"} {"<PcdsDynamicExVpd>"} <EOL>
                               [<DefineStatements>]*
                               [<SetStatements>]*
                               [<PcdsDynamicExEntry>]+

<BuildOptions>          ::=  "<BuildOptions>" <EOL>
                               [<DefineStatements>]*
                               [<ToolFlags>]+
                               [<ToolPath>]*
                               [<ToolCmd>]*
                               [<Other>]*

<ToolFlags>             ::=  [<Family> ":"] <FlagSpec> <Equal> <Flags> <EOL>

<ToolSpec>              ::=  <Target> "_" <TagName> "_" <Tarch> "_" <ToolCode>

<FlagSpec>              ::=  <ToolSpec> "_FLAGS"

<ToolPath>              ::=  [<Family> ":"] <PathSpec> <Equal> <PATH> <EOL>

<PathSpec>              ::=  <ToolSpec> "_DPATH"
```

```
<ToolCmd>          ::=  [<Family> ":"] <CmdSpec> "==" <PathCmd> <EOL>

<CmdSpec>          ::=  <ToolSpec> "_PATH"

<PathCmd>          ::=  <PATH> <Word> ["." <Extension>]*

<Extension>        ::=  [a-zA-Z][a-zA-Z0-9_-]*

<Other>            ::=  [<Family> ":"] <OtherSpec>

<OtherSpec>        ::=  <ToolSpec> "_" <Attribute> <Equal> <String>

<Equal>            ::=  {"="} {"=="}

<Tarch>            ::=  {"IA32"} {"X64"} {"IPF"} {"EBC"} {*} {<OA>}

<OA>               ::=  [A-Z][A-Z0-9]*

<Family>           ::=  {"MSFT"} {"GCC"} {"INTEL"} {"ACPICA"} {<UserDef>}

<UserDef>          ::=  [A-Z][A-Z0-9]

<Attribute>        ::=  Attribute

<Target>           ::=  Target

<TagName>          ::=  TagName

<ToolCode>         ::=  ToolCode

<Flags>            ::=  FlagString
```

Parameters

Target

Must match a target identifier in the EDK II tools_def.txt file - the first field, where fields are separated by the underscore character. The * character is a valid wild-card.

TagName

Must match a tag name field in the EDK II tools_def.txt file – second field. The "*" character is a valid wild-card.

TargetArch

Must match the architecture field in the EDK II tools_def.txt file – third field. The "*" character is a valid wild-card.

ToolCode

Must match a tool code field in the EDK II tools_def.txt file – fourth field. Use of a wild-card character is not permitted.

AttributeName

Must match a tool attribute field in the EDK II tools_def.txt file – fifth field. Use of a wild-card character is not permitted. The attributes, _Flag, _PATH and DPATH are defined elsewhere and cannot be used with the <OtherSpec> definition.

FlagString

Must be a valid string for the tool specified. The string will be appended to the end of the tool's flags (from the tools_def.txt). Both Microsoft and GCC evaluate options from left to right on the command line. This allows disabling some flags that may have been specified in the tools_def.txt by providing an alternate flag, i.e., if the tools_def.txt CC FLAGS defines /O2 and an /O1 options is specified for this module, the module will compile with /O1 (size) not with /O2 (speed). Use of the quote characters around options is required when specifying string values with spaces, path names with spaces or values containing the hash "#" character not within a string. Note that a macro named MDEPKG_NDEBUG is reserved for size reduction purposes. The user should not use this keyword to define new macro.

Pcd Values

PCD elements follow the exact format defined for **<PcdEntry>** elements in the PCD Sections.

Example Using EDK components in an EDK II DSC build

```
[Components]
DEFINE EDK=$(EDK_SOURCE)/Edk
DEFINE MDE=$(WORKSPACE)/MdePkg/Library
DEFINE STATUS_CODE=$(MDE)/PeiDxeDebugLibReportStatusCode

$(EDK)/Foundation/Core/Pei/PeiMain.inf
$(EDK)/Sample/Platform/Generic/MonoStatusCode/Pei/Nt32/MonoStatusCode.inf
$(EDK)/Sample/Platform/Nt32/Pei/BootMode/BootMode.inf
$(EDK)/Sample/Platform/Nt32/Pei/FlashMap/FlashMap.inf
$(WORKSPACE)/MdeModulePkg/Core/Dxe/DxeMain.inf...
$(WORKSPACE)/MdeModulePkg/Universal/Debugger/Debugport/Dxe/DebugPort.inf
$(WORKSPACE)/MdeModulePkg/Cpu/DebugSupport/Dxe/DebugSupport.inf
...
$(WORKSPACE)/MdeModulePkg/Universal/DataHub/DataHubStdErr/Dxe/DataHubStdErr.inf
$(WORKSPACE)/MdeModulePkg/Universal/Disk/DiskIo/Dxe/DiskIo.inf {
  <LibraryClasses>
    DebugLib|$(STATUS_CODE)/PeiDxeDebugLibReportStatusCode.inf
    BaseMemoryLib|$(MDE)/DxeMemoryLib/DxeMemoryLib.inf
    MemoryAllocationLib | \
      $(MDE)/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  <PcdsFeatureFlag>
    PcdDriverDiagnosticsDisable|gEfiMdePkgTokenSpaceGuid|FALSE
}
$(WORKSPACE)/MdeModulePkg/Universal/Ebc/Dxe/Ebc.inf
$(WORKSPACE)/MdeModulePkg/Universal/GenericMemoryTest/Dxe/NullMemoryTest.inf
$(WORKSPACE)/MdeModulePkg/Universal/StatusCode/Pei/PeiStatusCode.inf {
    <BuildOptions>
      MSFT:RELEASE_MYTOOLS_IA32_DLINK_FLAGS    = Kernel32.lib \
        MSVCRTD.lib Gdi32.lib User32.lib Winmm.lib
      DEBUG_MYTOOLS_IA32_DLINK_FLAGS           = Kernel32.lib \
        MSVCRTD.lib Gdi32.lib User32.lib Winmm.lib
      DEBUG_WINDDK3790x1830_IA32_DLINK_FLAGS   = Kernel32.lib \
        MSVCRTD.lib Gdi32.lib User32.lib Winmm.lib
      RELEASE_WINDDK3790x1830_IA32_DLINK_FLAGS = Kernel32.lib \
        MSVCRTD.lib Gdi32.lib User32.lib Winmm.lib
      DEBUG_VS2003_IA32_DLINK_FLAGS            = Kernel32.lib \
        MSVCRTD.lib Gdi32.lib User32.lib Winmm.lib
      RELEASE_VS2003_IA32_DLINK_FLAGS          = Kernel32.lib \
        MSVCRTD.lib Gdi32.lib User32.lib Winmm.lib
  }
$(WORKSPACE)/EdkNt32Pkg/Logo/Logo.inf
...
```

3.11 [UserExtensions] Section

Summary

Defines the optional EDK II DSC file `[UserExtensions]` section tag. This is an optional section.

There can be multiple UserExtension sections, depending on the UserID and Identifier attributes of the FPD file. An example section is listed below. UserExtensions are for processing by tools outside of the standard tools provided by EDK II.

The EDK II build tools will ignore this section.

The data in the DSC file must be entered by the platform developer, as no data will be copied from the FPD files. (Current FPD files use this section to control how binary files are put into flash images, which is outside the scope of the EDK II DSC files.)

Format for the section header is `[UserExtensions.UserId.Identifier]`

Prototype

```
<UserExtensions>    ::=  "[UserExtensions" <UserId> <IdentifierString> "]" <EOL>
                         [<statements>]

<UserId>            ::=  "." <Word>

<IdentifierString>  ::=  "." {<Word>} {<QuotedString>}

<statements>        ::=  Content is build tool chain specific.
```

Example

```
[UserExtensions.MyOrgDotCom.1.0]
```

Appendix A
Variables

One of the core concepts of this utility is the notion of symbols. Use of symbols follows the makefile convention of enclosing within $(), for example **$(EDK_SOURCE)**. As the parsing utility processes files, it will often perform parsing of variable assignments. These variables can then be referenced in other sections of the DSC file. Variable assignments will be saved internally in either a local or global symbol table. The local symbol table is purged following processing of individual Platform (DSC) files. Global symbol values persist throughout execution of the utility. Local symbol values take precedent over global symbols. The following table describes the symbols generated internally by the utility. They can be overridden either on the command line, in the DSC file, or in individual INF files. The symbols in the table below are typically global.

For a pure EDK II build, two environment variables are required to be set prior to executing any build, **WORKSPACE** must put to the root of the EDK II directory tree, while **EDK_TOOLS_PATH** must point to the Location of the BaseTools directory tree.

For a build using EDK components in the EDK II DSC file, the **EDK_SOURCE** directory must point to the root of the EDK (either a directory containing an EDK tree or the location of the EdkCompatibilityPackage directory) tree.

Table 5. Standard Variables

Variable Name	Description
`WORKSPACE`	Defines the root directory of the local EDK II source tree, for example C:\EDK2. If not defined as an environmental variable when an EDK II tool is invoked, the utility will give an error message and exit.
`EDK_TOOLS_PATH`	Defines the root directory of the local EDK II BaseTools directory. Default: `$(WORKSPACE)/BaseTools`
`EDK_SOURCE`	Defines the directory of an original EDK directory tree, for example C:\EFI2.0\EdkCompatibilityPkg. If not defined as an environmental variable when an EDK II tool is invoked, the utility will give an error message and exit.
`EFI_SOURCE`	Defines the root directory of a local original EDK source tree, for example C:\EFI2.0\MyEdkDriver.
`EFI_TOOLS_PATH`	Defines the root directory containing the original EDK Tools directory tree, for example C:\Tools.
`EFI_TOOLS_BIN`	Defines the directory containing the original EDK Tools executable files. Default: `$(EFI_TOOLS_PATH)/Bin`
`PROCESSOR`	Defines the EDK target processor for which the code is to be built. This symbol will typically be used only in EDK INF files, and is ignored for pure EDK II builds.
`TARGET_ARCH`	This defines which of the supported architectures will be built – this value is set in the file: `$(WORKSPACE)/Conf/target.txt` or it may be specified on a command-line.
`BUILD_DIR`	Defines the build tip directory for the current platform. For example, this may be `$(OUTPUT_DIR)\Platform\Nt32`.
`SOURCE_DIR`	For a component, defines the directory of the component source files.
`DEST_DIR`	For a component, defines the directory (typically under BUILD_DIR) where the component object files are to be built.
`LIB_DIR`	Specifies the directory where EFI libraries are deposited after building. Default: `$(BUILD_DIR)\$(TARGET)_$(TAGNAME)\$(ARCH)\LIB`
`BIN_DIR`	Specifies the directory where final component binaries (.efi) are deposited during build. Default: `$(BUILD_DIR)\$(TARGET)_$(TAGNAME)\$(ARCH)`
`FV_DIR`	Specifies a WORKSPACE relative or absolute directory where the final image files should be placed at the conclusion of a successful build. If the directory does not exist, the tools should create this directory. If not defined, tools should create an FV directory underneath the BIN_DIR.

DSC_FILENAME	Name of the DSC file as specified on the command line. Can be used for dependencies in the makefiles.
INF_FILENAME	Name of the INF file for a given EDK component or EDK II Module. Can be used for dependencies in the makefiles.
MAKEFILE_NAME	Name of the output makefile for the platform. Default is “makefile”. The default value can be overridden to support keeping multiple variations of a makefile in the same DEST_DIR directory.

Appendix B
Sample EDK II DSC File

The following EDK II DSC file is an example of the Nt32.dsc showing a typical “user” maintained version, which simplifies the structure over the previous example.

Note: *The "\" line continuation character is used here for clarity, but is not permitted in actual DSC files. Continued lines are indented to match the indentation of the beginning of the line.*

```
## @file
# EFI/Framework Emulation Platform
#
# The Emulation Platform can be used to debug individual modules,
# prior to creating a real platform. This also provides an
# example for how an DSC is created.
#
# Copyright (c) 2006 - 2007, Intel Corporation
#
#  All rights reserved. This program and the accompanying
#  materials are licensed and made available under the terms and
#  conditions of the BSD License which accompanies this
#  distribution. The full text of the license may be found at
#    http://opensource.org/licenses/bsd-license.php
#
#    THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS
#    IS" BASIS,
#    WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER
#    EXPRESS OR IMPLIED.
#
##


######################################################################
#
# Defines Section - statements that will be processed to create
#                   a Makefile.
######################################################################
#
#


[Defines]
  PLATFORM_NAME                  = NT32
  PLATFORM_GUID                  = EB216561-961F-47EE-9EF9-CA426EF547C2
  PLATFORM_VERSION               = 0.3
  DSC_SPECIFICATION              = 0x00010015
  OUTPUT_DIRECTORY               = Build/NT32
  SUPPORTED_ARCHITECTURES        = IA32
  BUILD_TARGETS                  = DEBUG
  SKUID_IDENTIFIER               = DEFAULT
  FLASH_DEFINITION               = Nt32Pkg/Nt32Pkg.fdf


######################################################################
#
# SKU Identification section - list of all SKU IDs supported
#                              by this Platform.
#
######################################################################
[SkuIds]
  0|DEFAULT              # The entry: 0|DEFAULT is reserved and always required.
```

```
##################################################################
# Library Class section - list of all Library Classes needed by
#                         this Platform.
#
##################################################################

[LibraryClasses.common]
  TimerLib|MdePkg/Library/BaseTimerLibNullTemplate/BaseTimerLibNullTemplate.inf
  PrintLib|MdePkg/Library/BasePrintLib/BasePrintLib.inf
  SerialPortLib|MdePkg/Library/SerialPortLibNull/SerialPortLibNull.inf
  BaseMemoryLib|MdePkg/Library/BaseMemoryLib/BaseMemoryLib.inf
  BaseLib|MdePkg/Library/BaseLib/BaseLib.inf
  PerformanceLib|MdePkg/Library/BasePerformanceLibNull/BasePerformanceLibNull.inf
  PeCoffLib|MdePkg/Library/BasePeCoffLib/BasePeCoffLib.inf
  PciIncompatibleDeviceSupportLib|\
  IntelFrameworkModulePkg/Library/PciIncompatibleDeviceSupportLib/\
  PciIncompatibleDeviceSupportLib.inf
  CacheMaintenanceLib|MdePkg/Library/BaseCacheMaintenanceLib/BaseCacheMaintenanceLib.inf
  IfrSupportLibFramework|IntelFrameworkPkg/Library/IfrSupportLibFramework/IfrSupportLib.inf
  GraphicsLib|IntelFrameworkModulePkg/Library/GraphicsLib/GraphicsLib.inf
  FvbServiceLib|MdeModulePkg/Library/EdkFvbServiceLib/EdkFvbServiceLib.inf
  IoLib|MdePkg/Library/BaseIoLibIntrinsic/BaseIoLibIntrinsic.inf
  CustomDecompressLib|\
  IntelFrameworkModulePkg/Library/BaseUefiTianoCustomDecompressLib/\
  BaseUefiTianoCustomDecompressLib.inf
  UefiDecompressLib|\
  IntelFrameworkModulePkg/Library/BaseUefiTianoCustomDecompressLib/\
  BaseUefiTianoCustomDecompressLib.inf
  HiiLibFramework|IntelFrameworkPkg/Library/HiiLibFramework/HiiLib.inf

[LibraryClasses.common.BASE]
  DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/BaseReportStatusCodeLib/BaseReportStatusCodeLib.inf

[LibraryClasses.common.USER_DEFINED]
  DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/BaseReportStatusCodeLib/BaseReportStatusCodeLib.inf

[LibraryClasses.common.SEC]
  DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/BaseReportStatusCodeLib/BaseReportStatusCodeLib.inf

[LibraryClasses.common.DXE_CORE]
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf
  UefiBootServicesTableLib|\
  MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib.inf
  HobLib|MdePkg/Library/DxeCoreHobLib/DxeCoreHobLib.inf
  DxeCoreEntryPoint|MdePkg/Library/DxeCoreEntryPoint/DxeCoreEntryPoint.inf
  MemoryAllocationLib|MdePkg/Library/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/DxeReportStatusCodeLibFramework/DxeReportStatusCodeLib.inf
```

```
  PeCoffLoaderLib|\
  MdeModulePkg/Library/DxePeCoffLoaderFromHobLib/DxePeCoffLoaderFromHobLib.inf
  UefiRuntimeServicesTableLib|\
  MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServicesTableLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf


[LibraryClasses.common.DXE_SMM_DRIVER]
  DxeServicesTableLib|MdePkg/Library/DxeServicesTableLib/DxeServicesTableLib.inf
  ScsiLib|MdePkg/Library/UefiScsiLib/UefiScsiLib.inf
  HiiLibFramework|IntelFrameworkPkg/Library/HiiLibFramework/HiiLib.inf
  UefiBootServicesTableLib|\
  MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib.inf
  HobLib|MdePkg/Library/DxeHobLib/DxeHobLib.inf
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf
  PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
  MemoryAllocationLib|MdePkg/Library/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/DxeReportStatusCodeLibFramework/DxeReportStatusCodeLib.inf
  UefiRuntimeServicesTableLib|\
  MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServicesTableLib.inf
  WinNtLib|Nt32Pkg/Library/DxeWinNtLib/DxeWinNtLib.inf
  OemHookStatusCodeLib|\
  Nt32Pkg/Library/DxeNt32OemHookStatusCodeLib/DxeNt32OemHookStatusCodeLib.inf
  PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
  DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf

[LibraryClasses.common.PEIM]
  HobLib|MdePkg/Library/PeiHobLib/PeiHobLib.inf
  PcdLib|MdePkg/Library/PeiPcdLib/PeiPcdLib.inf
  IoLib|MdePkg/Library/PeiIoLibCpuIo/PeiIoLibCpuIo.inf
  PeimEntryPoint|MdePkg/Library/PeimEntryPoint/PeimEntryPoint.inf
  PeiServicesLib|MdePkg/Library/PeiServicesLib/PeiServicesLib.inf
  MemoryAllocationLib|MdePkg/Library/PeiMemoryAllocationLib/PeiMemoryAllocationLib.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/PeiReportStatusCodeLib/PeiReportStatusCodeLib.inf
  PeCoffLoaderLib|Nt32Pkg/Library/Nt32PeCoffLoaderLib/Nt32PeCoffLoaderLib.inf
  PeiServicesTablePointerLib|\
  MdePkg/Library/PeiServicesTablePointerLib/PeiServicesTablePointerLib.inf
  OemHookStatusCodeLib|\
  Nt32Pkg/Library/PeiNt32OemHookStatusCodeLib/PeiNt32OemHookStatusCodeLib.inf
  PeCoffGetEntryPointLib|\
  Nt32Pkg/Library/Nt32PeiPeCoffGetEntryPointLib/Nt32PeiPeCoffGetEntryPointLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf

[LibraryClasses.common.PEI_CORE]
  HobLib|MdePkg/Library/PeiHobLib/PeiHobLib.inf
  PeiServicesTablePointerLib|\
  MdePkg/Library/PeiServicesTablePointerLib/PeiServicesTablePointerLib.inf
  IoLib|MdePkg/Library/PeiIoLibCpuIo/PeiIoLibCpuIo.inf
```

```
  PeiServicesLib|MdePkg/Library/PeiServicesLib/PeiServicesLib.inf
  MemoryAllocationLib|MdePkg/Library/PeiMemoryAllocationLib/PeiMemoryAllocationLib.inf
  PeiCoreEntryPoint|MdePkg/Library/PeiCoreEntryPoint/PeiCoreEntryPoint.inf
  OldPeiCoreEntryPoint|MdePkg/Library/OldPeiCoreEntryPoint/OldPeiCoreEntryPoint.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/PeiReportStatusCodeLib/PeiReportStatusCodeLib.inf
  PeCoffGetEntryPointLib|\
  Nt32Pkg/Library/Nt32PeiPeCoffGetEntryPointLib/Nt32PeiPeCoffGetEntryPointLib.inf
  PcdLib|MdePkg/Library/PeiPcdLib/PeiPcdLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf

[LibraryClasses.common.DXE_RUNTIME_DRIVER]
  UefiRuntimeServicesTableLib|\
  MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServicesTableLib.inf
  HobLib|MdePkg/Library/DxeHobLib/DxeHobLib.inf
  PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
  MemoryAllocationLib|MdePkg/Library/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  UefiDriverEntryPoint|MdePkg/Library/UefiDriverEntryPoint/UefiDriverEntryPoint.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/DxeReportStatusCodeLibFramework/DxeReportStatusCodeLib.inf
  DxeServicesTableLib|MdePkg/Library/DxeServicesTableLib/DxeServicesTableLib.inf
  HiiLibFramework|IntelFrameworkPkg/Library/HiiLibFramework/HiiLib.inf
  UefiBootServicesTableLib|\
  MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib.inf
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf
  UefiRuntimeLib|MdePkg/Library/UefiRuntimeLib/UefiRuntimeLib.inf
  ScsiLib|MdePkg/Library/UefiScsiLib/UefiScsiLib.inf
  WinNtLib|Nt32Pkg/Library/DxeWinNtLib/DxeWinNtLib.inf
  OemHookStatusCodeLib|\
  Nt32Pkg/Library/DxeNt32OemHookStatusCodeLib/DxeNt32OemHookStatusCodeLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf

[LibraryClasses.common.UEFI_DRIVER]
  UefiRuntimeServicesTableLib|\
  MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServicesTableLib.inf
  HobLib|MdePkg/Library/DxeHobLib/DxeHobLib.inf
  PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
  MemoryAllocationLib|MdePkg/Library/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  UefiDriverEntryPoint|MdePkg/Library/UefiDriverEntryPoint/UefiDriverEntryPoint.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/DxeReportStatusCodeLibFramework/DxeReportStatusCodeLib.inf
  DxeServicesTableLib|MdePkg/Library/DxeServicesTableLib/DxeServicesTableLib.inf
  HiiLibFramework|IntelFrameworkPkg/Library/HiiLibFramework/HiiLib.inf
  UefiBootServicesTableLib|\
  MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib.inf
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf
  ScsiLib|MdePkg/Library/UefiScsiLib/UefiScsiLib.inf
  WinNtLib|Nt32Pkg/Library/DxeWinNtLib/DxeWinNtLib.inf
  OemHookStatusCodeLib|\
```

```
  Nt32Pkg/Library/DxeNt32OemHookStatusCodeLib/DxeNt32OemHookStatusCodeLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf

[LibraryClasses.common.DXE_DRIVER]
  UefiRuntimeServicesTableLib|\
  MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServicesTableLib.inf
  HobLib|MdePkg/Library/DxeHobLib/DxeHobLib.inf
  PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
  MemoryAllocationLib|MdePkg/Library/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  UefiDriverEntryPoint|MdePkg/Library/UefiDriverEntryPoint/UefiDriverEntryPoint.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/DxeReportStatusCodeLibFramework/DxeReportStatusCodeLib.inf
  DxeServicesTableLib|MdePkg/Library/DxeServicesTableLib/DxeServicesTableLib.inf
  HiiLibFramework|IntelFrameworkPkg/Library/HiiLibFramework/HiiLib.inf
  UefiBootServicesTableLib|\
  MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib.inf
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf
  ScsiLib|MdePkg/Library/UefiScsiLib/UefiScsiLib.inf
  WinNtLib|Nt32Pkg/Library/DxeWinNtLib/DxeWinNtLib.inf
  OemHookStatusCodeLib|\
  Nt32Pkg/Library/DxeNt32OemHookStatusCodeLib/DxeNt32OemHookStatusCodeLib.inf
  EdkGenericBdsLib|Nt32Pkg/Library/EdkGenericBdsLib/EdkGenericBdsLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf

[LibraryClasses.common.UEFI_APPLICATION]
  DxeServicesTableLib|MdePkg/Library/DxeServicesTableLib/DxeServicesTableLib.inf
  ScsiLib|MdePkg/Library/UefiScsiLib/UefiScsiLib.inf
  HiiLibFramework|IntelFrameworkPkg/Library/HiiLibFramework/HiiLib.inf
  UefiApplicationEntryPoint|\
  MdePkg/Library/UefiApplicationEntryPoint/UefiApplicationEntryPoint.inf
  UefiBootServicesTableLib|\
  MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib.inf
  HobLib|MdePkg/Library/DxeHobLib/DxeHobLib.inf
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf
  PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
  MemoryAllocationLib|MdePkg/Library/DxeMemoryAllocationLib/DxeMemoryAllocationLib.inf
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  ReportStatusCodeLib|\
  IntelFrameworkModulePkg/Library/DxeReportStatusCodeLibFramework/DxeReportStatusCodeLib.inf
  UefiRuntimeServicesTableLib|\
  MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServicesTableLib.inf
  WinNtLib|Nt32Pkg/Library/DxeWinNtLib/DxeWinNtLib.inf
  OemHookStatusCodeLib|\
  Nt32Pkg/Library/DxeNt32OemHookStatusCodeLib/DxeNt32OemHookStatusCodeLib.inf
  DebugLib|\
  IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/\
  PeiDxeDebugLibReportStatusCode.inf


##################################################################
```

```
#
# Pcd Section - list of all EDK II PCD Entries defined by this
#               Platform
#
#############################################################################

[PcdsFeatureFlag.common]
  gEfiMdeModulePkgTokenSpaceGuid.PcdDevicePathSupportDevicePathFromText|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDevicePathSupportDevicePathToText|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportCustomDecompress|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplBuildShareCodeHobs|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportEfiDecompress|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportTianoDecompress|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportCustomDecompress|TRUE
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultLangDepricate|FALSE

[PcdsFixedAtBuild.IA32]
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeMemorySize|1
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeRuntimeMemorySize.|128
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtMemorySizeForSecMain|L"64!64"|VOID*|12
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFirmwareVolume|L"..\\Fv\\Fv_Recovery.fd"|VOID*|52
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtBootMode|1
  gEfiMdePkgTokenSpaceGuid.PcdMaximumUnicodeStringLength|1000000
  gEfiMdePkgTokenSpaceGuid.PcdMaximumAsciiStringLength|1000000
  gEfiMdePkgTokenSpaceGuid.PcdMaximumLinkedListLength|1000000
  gEfiMdePkgTokenSpaceGuid.PcdSpinLockTimeout|10000000
  gEfiMdePkgTokenSpaceGuid.PcdMaximumAsciiStringLength|1000000
  gEfiMdePkgTokenSpaceGuid.PcdMaximumLinkedListLength|1000000
  gEfiMdePkgTokenSpaceGuid.PcdSpinLockTimeout|10000000
  gEfiMdePkgTokenSpaceGuid.PcdReportStatusCodePropertyMask|0x0f
  gEfiMdePkgTokenSpaceGuid.PcdDebugPropertyMask|0x1f
  gEfiMdePkgTokenSpaceGuid.PcdDebugClearMemoryValue|0xAF
  gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000040
  gEfiMdePkgTokenSpaceGuid.PcdPerformanceLibraryPropertyMask|0
  gEfiMdeModulePkgTokenSpaceGuid.PcdMaxPeiPcdCallBackNumberPerPcdEntry|0x08
  gEfiMdeModulePkgTokenSpaceGuid.PcdVpdBaseAddress|0x0
  gEfiMdeModulePkgTokenSpaceGuid.PcdMaxSizeNonPopulateCapsule|0x0
  gEfiMdeModulePkgTokenSpaceGuid.PcdMaxSizePopulateCapsule|0x0
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPciIncompatibleDeviceSupportMask|0
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueUncorrectableMemoryError|\
  0x0005100   # EFI_COMPUTING_UNIT_MEMORY | EFI_CU_MEMORY_EC_UNCORRECTABLE3
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueRemoteConsoleError|\
  0x01040006        # EFI_PERIPHERAL_REMOTE_CONSOLE | EFI_P_EC_CONTROLLER_ERROR
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueRemoteConsoleReset|\
  0x01040001        # EFI_PERIPHERAL_REMOTE_CONSOLE | EFI_P_PC_RESET
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueRemoteConsoleInputError|\
  0x01040007   # EFI_PERIPHERAL_REMOTE_CONSOLE | EFI_P_EC_INPUT_ERROR
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueRemoteConsoleOutputError|\
  0x01040008  # EFI_PERIPHERAL_REMOTE_CONSOLE | EFI_P_EC_OUTPUT_ERROR
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultTimeout|0x0008
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultLangCodes|"engfra"|7
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultLang|"eng"|VOID*|4
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultPlatformLangCodes|"en;fr"|VOID*|6
  gEfiMdePkgTokenSpaceGuid.PcdUefiVariableDefaultPlatformLang|"en"|VOID*|2
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueEfiWatchDogTimerExpired|0x00011003
```

```
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueMemoryTestStarted|0x00051006
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueSetVirtualAddressMap|0x03101004
  gEfiMdePkgTokenSpaceGuid.PcdStatusCodeValueUncorrectableMemoryError|0x00051003
  gEfiMdePkgTokenSpaceGuid.PcdUefiLibMaxPrintBufferSize|320

  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFirmwareFdSize|0x2a0000
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFirmwareBlockSize|0x10000
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashNvStorageEventLogBase|0x28c000
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashNvStorageEventLogSize|0x2000
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashFvRecoveryBase|0x0
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashFvRecoverySize|0x280000

  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashNvStorageVariableBase|0x280000
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashNvStorageFtwSpareBase|0x290000
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtFlashNvStorageFtwWorkingBase|0x28e000

  gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageFtwSpareSize|0x10000
  gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageFtwWorkingSize|0x2000
  gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageVariableSize|0x00c000

[PcdsFeatureFlag.IA32]
  gEfiMdeModulePkgTokenSpaceGuid.PcdPeiPcdDatabaseTraverseEnabled|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdPeiPcdDatabaseCallbackOnSetEnabled|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdPeiPcdDatabaseExEnabled|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdPeiPcdDatabaseGetSizeEnabled|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdPeiPcdDatabaseSetEnabled|TRUE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseSerial|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseMemory|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseOEM|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSwitchToLongMode|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportEfiDecompress|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportTianoDecompress|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplSupportCustomDecompress|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxeIplBuildShareCodeHobs|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDxePcdDatabaseTraverseEnabled|TRUE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseHardSerial|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseEfiSerial|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseRuntimeMemory|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseDataHub|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeUseOEM|TRUE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeReplayInSerial|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeReplayInDataHub|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeReplayInRuntimeMemory|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdStatusCodeReplayInOEM|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdSupportUpdateCapsuleRest|FALSE
  gEfiMdePkgTokenSpaceGuid.PcdComponentNameDisable|FALSE
  gEfiMdePkgTokenSpaceGuid.PcdDriverDiagnosticsDisable|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdNtEmulatorEnable|FALSE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDevicePathSupportDevicePathToText|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdDevicePathSupportDevicePathFromText|TRUE
  gEfiMdePkgTokenSpaceGuid.PcdComponentName2Disable|TRUE
  gEfiMdePkgTokenSpaceGuid.PcdDriverDiagnostics2Disable|TRUE
  gEfiMdeModulePkgTokenSpaceGuid.PcdSupportUpdateCapsuleRest|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPciIsaEnable|FALSE
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPciVgaEnable|FALSE
```

```
  gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPciBusHotplugDeviceSupport|TRUE


######################################################################
#
# Pcd Dynamic Section - list of all EDK II PCD Entries defined by
#                       this Platform
#
######################################################################

[PcdsDynamicDefault.common.DEFAULT]
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtSerialPort|L"COM1!COM2"|VOID*|18
  PcdWinNtFileSystem|gEfiNt32PkgTokenSpaceGuid|\
  L".!..\\..\\..\\..\\EdkShellBinPkg\\bin\\ia32\\Apps"|106
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtGop|L"UGA Window 1!UGA Window 2"|VOID*|50
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtConsole|L"Bus Driver Console Window"|VOID*|50
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtVirtualDisk|L"FW;40960;512"|VOID*|24
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtCpuModel|L"NT32 Processor Emulation"|VOID*|52
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtCpuSpeed|L"1234"|VOID*|8
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtMemorySize|L"64!64"|VOID*|10
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtPhysicalDisk|L"a:RW;2880;512!e:RW;262144;512"|VOID*|58
  gEfiNt32PkgTokenSpaceGuid.PcdWinNtUga|L"UGA Window 1!UGA Window 2"|VOID*|50

  gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageFtwSpareBase|0
  gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageFtwWorkingBase|0
  gEfiMdeModulePkgTokenSpaceGuid.PcdFlashNvStorageVariableBase|0

######################################################################
#
# Components Section - list of all EDK II Modules needed by
#                      this Platform
#
######################################################################

[Components.IA32]
  ##
  #  SEC Phase modules
  ##
  Nt32Pkg/Sec/SecMain.inf

  ##
  #  PEI Phase modules
  ##
  MdeModulePkg/Core/Pei/PeiMain.inf
  MdeModulePkg/Universal/PCD/Pei/Pcd.inf  {
   <LibraryClass>
      PcdLib|MdePkg/Library/BasePcdLibNull/BasePcdLibNull.inf
  }
  IntelFrameworkModulePkg/Universal/StatusCode/Pei/PeiStatusCode.inf
  Nt32Pkg/BootModePei/BootModePei.inf
  Nt32Pkg/WinNtFlashMapPei/WinNtFlashMapPei.inf
  MdeModulePkg/Universal/MemoryTest/BaseMemoryTestPei/BaseMemoryTestPei.inf
  MdeModulePkg/Universal/Variable/Pei/VariablePei.inf
  Nt32Pkg/WinNtAutoScanPei/WinNtAutoScanPei.inf
  Nt32Pkg/WinNtFirmwareVolumePei/WinNtFirmwareVolumePei.inf
```

```
Nt32Pkg/WinNtThunkPPIToProtocolPei/WinNtThunkPPIToProtocolPei.inf
MdeModulePkg/Core/DxeIplPeim/DxeIpl.inf

##
#  DXE Phase modules
##
MdeModulePkg/Core/Dxe/DxeMain.inf
MdeModulePkg/Universal/PCD/Dxe/Pcd.inf {
  <LibraryClass>
    PcdLib|MdeModulePkg/Library/PcdDriverPcdLibNull/PcdLib.inf
}
Nt32Pkg/MetronomeDxe/MetronomeDxe.inf
Nt32Pkg/RealTimeClockRuntimeDxe/RealTimeClockRuntimeDxe.inf
Nt32Pkg/ResetRuntimeDxe/ResetRuntimeDxe.inf
MdeModulePkg/Core/RuntimeDxe/RuntimeDxe.inf
Nt32Pkg/FvbServicesRuntimeDxe/FvbServicesRuntimeDxe.inf
MdeModulePkg/Universal/SecurityStubDxe/SecurityStubDxe.inf
IntelFrameworkModulePkg/Universal/DataHubDxe/DataHubDxe.inf
MdeModulePkg/Universal/EbcDxe/EbcDxe.inf
MdeModulePkg/Universal/MemoryTest/NullMemoryTestDxe/NullMemoryTestDxe.inf
IntelFrameworkModulePkg/Universal/HiiDataBaseDxe/HiiDatabase.inf
Nt32Pkg/WinNtThunkDxe/WinNtThunkDxe.inf
MdeModulePkg/Universal/FirmwareVolume/Crc32SectionExtractDxe/Crc32SectionExtractDxe.inf
Nt32Pkg/CpuRuntimeDxe/CpuRuntimeDxe.inf
Nt32Pkg/PlatformBdsDxe/PlatformBdsDxe.inf
MdeModulePkg/Universal/FirmwareVolume/FaultTolerantWriteDxe/FtwLite.inf
IntelFrameworkModulePkg/Universal/DataHubStdErrDxe/DataHubStdErrDxe.inf
Nt32Pkg/MiscSubClassPlatformDxe/MiscSubClassPlatformDxe.inf
Nt32Pkg/TimerDxe/TimerDxe.inf
IntelFrameworkModulePkg/Universal/StatusCode/Dxe/DxeStatusCode.inf
MdeModulePkg/Universal/Variable/RuntimeDxe/VariableRuntimeDxe.inf
MdeModulePkg/Universal/WatchDogTimerDxe/WatchDogTimer.inf
MdeModulePkg/Universal/MonotonicCounterRuntimeDxe/MonotonicCounterRuntimeDxe.inf
MdeModulePkg/Universal/CapsuleRuntimeDxe/CapsuleRuntimeDxe.inf
MdeModulePkg/Universal/Console/ConPlatformDxe/ConPlatformDxe.inf
MdeModulePkg/Universal/Console/ConSplitterDxe/ConSplitterDxe.inf
MdeModulePkg/Universal/Console/GraphicsConsoleDxe/GraphicsConsoleDxe.inf
MdeModulePkg/Universal/Console/TerminalDxe/TerminalDxe.inf
MdeModulePkg/Universal/DevicePathDxe/DevicePathDxe.inf
MdeModulePkg/Universal/Disk/DiskIoDxe/DiskIoDxe.inf
MdeModulePkg/Universal/Disk/PartitionDxe/PartitionDxe.inf
IntelFrameworkModulePkg/Universal/SetupBrowserDxe/SetupBrowser.inf
MdeModulePkg/Universal/Disk/UnicodeCollation/EnglishDxe/EnglishDxe.inf
MdeModulePkg/Bus/Pci/AtapiPassThruDxe/AtapiPassThru.inf
IntelFrameworkModulePkg/Bus/Pci/PciBusDxe/PciBusDxe.inf
MdeModulePkg/Bus/Scsi/ScsiBusDxe/ScsiBusDxe.inf     \
##This driver follows UEFI specification definition
MdeModulePkg/Bus/Scsi/ScsiDiskDxe/ScsiDiskDxe.inf    \
##This driver follows UEFI specification definition
IntelFrameworkModulePkg/Bus/Pci/IdeBusDxe/IdeBusDxe.inf
Nt32Pkg/WinNtBusDriverDxe/WinNtBusDriverDxe.inf
Nt32Pkg/WinNtBlockIoDxe/WinNtBlockIoDxe.inf
Nt32Pkg/WinNtConsoleDxe/WinNtConsoleDxe.inf
Nt32Pkg/WinNtSerialIoDxe/WinNtSerialIoDxe.inf
Nt32Pkg/WinNtGopDxe/WinNtGopDxe.inf
```

```
  Nt32Pkg/WinNtSimpleFileSystemDxe/WinNtSimpleFileSystemDxe.inf
  IntelFrameworkModulePkg/Universal/DriverSampleDxe/DriverSampleDxe.inf
  MdeModulePkg/Application/HelloWorld/HelloWorld.inf

[BuildOptions]
  MSFT:DEBUG_*_IA32_DLINK_FLAGS = /EXPORT:InitializeDriver=_ModuleEntryPoint /ALIGN:4096 /
SUBSYSTEM:CONSOLE
  MSFT:RELEASE_*_IA32_DLINK_FLAGS = /ALIGN:4096

  DEBUG_ICC_IA32_DLINK_FLAGS                  = /EXPORT:InitializeDriver=_ModuleEntryPoint /
ALIGN:4096 /SUBSYSTEM:CONSOLE
  DEBUG_VS2003_IA32_DLINK_FLAGS               = /EXPORT:InitializeDriver=_ModuleEntryPoint /
ALIGN:4096 /SUBSYSTEM:CONSOLE
  DEBUG_MYTOOLS_IA32_DLINK_FLAGS              = /EXPORT:InitializeDriver=_ModuleEntryPoint /
ALIGN:4096 /SUBSYSTEM:CONSOLE
  DEBUG_WINDDK3790x1830_IA32_DLINK_FLAGS      = /EXPORT:InitializeDriver=_ModuleEntryPoint /
ALIGN:4096 /SUBSYSTEM:CONSOLE
  DEBUG_VS2005PRO_IA32_DLINK_FLAGS            = /EXPORT:InitializeDriver=_ModuleEntryPoint /
ALIGN:4096 /SUBSYSTEM:CONSOLE
  DEBUG_MIXED_IA32_DLINK_FLAGS                = /EXPORT:InitializeDriver=_ModuleEntryPoint /
ALIGN:4096 /SUBSYSTEM:CONSOLE
  RELEASE_ICC_IA32_DLINK_FLAGS                = /ALIGN:4096
  RELEASE_VS2003_IA32_DLINK_FLAGS             = /ALIGN:4096
  RELEASE_MYTOOLS_IA32_DLINK_FLAGS            = /ALIGN:4096
  RELEASE_WINDDK3790x1830_IA32_DLINK_FLAGS    = /ALIGN:4096
  RELEASE_VS2005PRO_IA32_DLINK_FLAGS          = /ALIGN:4096
  RELEASE_MIXED_IA32_DLINK_FLAGS              = /ALIGN:4096
```

Appendix C
Module Types

Table 6. EDK II Module Types

MODULE_TYPE	Supported Architecture Types	Description
BASE	Any	Modules or Libraries can be ported to any execution environment. This module type is intended to be used by silicon module developers to produce source code that is not tied to any specific execution environment.
SEC	Any	Modules of this type are designed to start execution at the reset vector of a CPU. They are responsible for preparing the platform for the PEI phase.
PEI_CORE	Any	This module type is used by PEI Core implementations that are compliant with the PI 1.2 Specification.
PEIM	Any	This module type is used by PEIMs that are compliant with PI 1.2.
DXE_CORE	Any	This module type is used by DXE Core implementations that are compliant with the PI 1.2 Specification.
DXE_DRIVER	Any	This module type is used by DXE Drivers that are compliant with the PI 1.2 Specification.
DXE_RUNTIME_DRIVER	Any	This module type is used by DXE Drivers that are compliant to the PI 1.2 Specification. These modules execute in both boot services and runtime services environments.
DXE_SAL_DRIVER	IPF	This module type is used by DXE Drivers that can be called in physical mode before SetVirtualAddressMap() is called and either physical mode or virtual mode after SetVirtualAddressMap() has been called. This module type is only available for IPF processor types.
DXE_SMM_DRIVER	IA32, X64	This module type is used by DXE Drivers that are loaded into SMRAM.
SMM_CORE	Any	This is the SMM core.
UEFI_DRIVER	Any	This module type is used by UEFI Drivers that are compliant with the EFI 1.10 and UEFI 2.0 specifications. These modules provide services in the boot services execution environment. UEFI Drivers that return EFI_SUCCESS are not unloaded from memory. UEFI Drivers that return an error are unloaded from memory.
UEFI_APPLICATION	Any	This module type is used by UEFI Applications that are compliant with the EFI 1.10 and EFI 2.0 specifications. UEFI Applications are always unloaded when they exit.